DET NATUR- OG BIOVIDENSKABELIGE FAKULTET
KØBENHAVNS UNIVERSITET

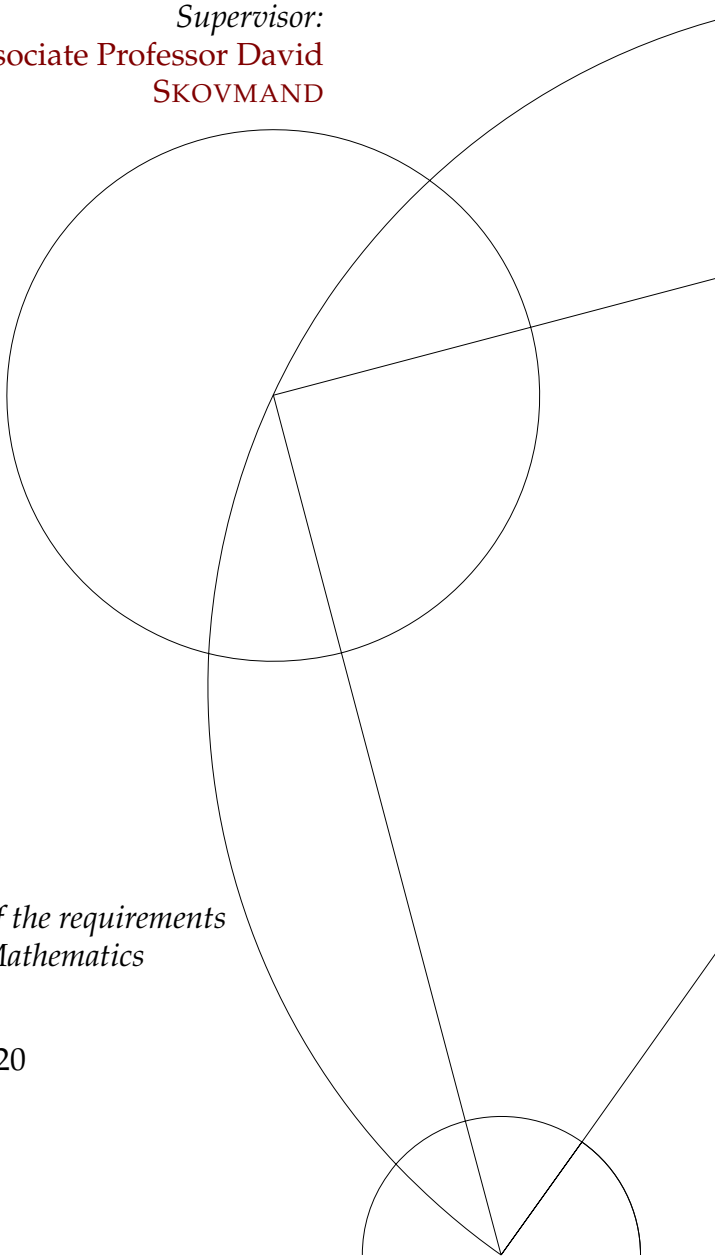
UNIVERSITY OF COPENHAGEN

MASTER'S THESIS

Classical Option Pricing Theory and Extensions to Deep Learning

Author:
Peter Pommergård LIND

Supervisor:
Associate Professor David SKOVMAND



A thesis submitted in fulfillment of the requirements for the degree of Actuarial Mathematics

November 12, 2020

Declaration of Authorship

I, Peter Pommergård LIND, declare that this thesis titled, "Classical Option Pricing Theory and Extensions to Deep Learning" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"You were hired because you met expectations, you will be promoted if you can exceed them."

Saji Ijiyemi

UNIVERSITY OF COPENHAGEN

Abstract

Department of Mathematical Science
Science

Actuarial Mathematics

Classical Option Pricing Theory and Extensions to Deep Learning

by Peter Pommergård LIND

The concepts for option pricing theory are presented and closed form solutions are provided in special cases. The options with no closed form solution are investigated through numerical methods, where both the binomial lattice model and LSM will be presented assuming the underlying Black-Scholes theory. Deep learning is then investigated to look for improvement of the existing methods, where we look specifically at the MLP regression. Our numerical study did not find any improvement in using MLP I instead of LSM. The MLP II was very fast, but lack the accuracy of the classical methods. Therefore, the MLP II could be beneficial in some circumstances where speed weighs more than precision. Even though the low accuracy for both deep learning methods, we believe MLP I and II can become an alternative for the classical methods with further investigation.

Acknowledgements

I would first like to thank my thesis advisor Associate Professor David Skovmand of the Department of Mathematical Science at University of Copenhagen. Prof. Skovmand consistently allowed this paper to be my own work, but steered me in the right direction whenever he thought I needed it.

I would also like to thank Joakim Pagels and Emil Petersen for providing useful discussions in the process. A special thanks goes to senior lecturer Mark Laplante at the University of Wisconsin Madison for being such an inspiring lecturer and discovering my interest in finance.

Finally, I must express my very profound gratitude to my parents and my family and friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Peter Pommergård Lind

Contents

List of Figures

List of Tables

List of Abbreviations

B-S	**B**lack-**S**choles
FPT1	**F**undamental **P**ricing **T**heorem **I**
FPT2	**F**undamental **P**ricing **T**heorem **II**
GBM	**G**eometric **B**rownian **M**otion
ITM	**I**n **T**he **M**oney
LIBOR	**L**ondon **I**nter**b**ank **O**ffered **R**ate
LSM	**L**east **S**quares Monte Carlo **M**ethod
MLP	**M**ulti**L**ayer **P**erceptron
MRT	**M**artingale **R**epresentation **T**heorem
PDE	**P**artial **D**ifferential **E**quation
RNVF	**R**isk **N**eutral **V**aluation **F**ormula
SDE	**S**tochastic **D**ifferential **E**quation
S-F	**S**elf-**F**inancing

Notations

Financial notation

c	European call option price
C	American call option price
p	European put option price
P	American put option price
K	Strike price
T	Maturity in years
σ	Volatility of asset
$S(0)$	Spot price
$S(T)$	Stock price at maturity
$S_i(t)$	i'th stock price at time t
r	Continuous compounding risk-free yearly interest rate
$V^h(\cdot)$	Value process of portfolio h
X	Simple Derivative
$\Phi(\cdot)$	Contract function
p_{ij}	Correlation coefficient between asset i and j
α_i	drift of the continuous lognormal distribution of asset i
$F(t, S(t))$	pricing function of derivative depending on S(t) and time t
d	number of risky assets

Mathematical symbols

$\mathbf{A}$	matrix notation for matrix A
$\mathbf{A}_{:,i}$	i'th column of matrix A
$\boldsymbol{a}$	vector notation for vector a
W_t	Weiner process under martingale measure Q
$\bar{W}_t$	Weiner process under probabilty measure P
$\mathbb{N}$	natural numbers: $1, 2, \ldots$
$\mathbb{R}$	real numbers
$\mathbb{R}^+$	real positive numbers including 0
$\mathbb{R}^+_*$	real positive numbers excluding 0
$\sim \mathcal{N}(\mu, \sigma^2)$	Normal distributed with mean μ and variance σ^2

Organization examples

Chapter 2	All the references are interactive marked with red lettering
Section 2.1	Under each chapter there are several sections
(Björk, 2009)	References to litteratur are in parenthesis
Equation (2.5)	Referred equations are in parenthesis
Link tensorboard 1	Links to websites are interactive marked with darkred lettering

Chapter 1

Introduction

The theory of option pricing dates back to Louis Bachelier with his PhD thesis *"The Theory of Speculation"* in 1900, but it was much later that option theory gained significant attention. In 1973 the world's first option exchange in Chicago opened, and in the same year Fisher Black and Myron Scholes came out with the first analytical formula for the European call option (Black and Scholes, 1973). This revolutionized market practice and option pricing theory. The idea of replication was born and the financial derivatives could now be priced by rational pricing.

The Black-Scholes model for European options is still used today, but the analytical framework cannot handle more complex products such as American put options. Since then we have seen an increasing complexity of financial products, where big investment banks have increased their need for financial engineers to handle the derivative books and price derivative products. With the complexity, a lot of challenges have arisen in this field, where a great understanding of the products is required to handle big derivative books. Most of the existing derivatives do not have a closed form solution, so numerical methods are used to approximate the price function. To emphasize the risk of derivatives without great understanding the successful trader Warren Buffett says derivatives are *"Financial weapons of mass destruction"* (p. 15 (Buffett, 2002)), but on the other hand, Warren Buffett has derivatives in his portfolio. A recent example of bad management of a derivative book is AIG, where AIG needed a bailout by the US government under the recent financial crisis in 2007-2009 (McDonald and Paulson, 2015). To sum up, derivatives give the trader more options either to utilize arbitrage, speculate, or hedge, but without care or knowledge about a derivative book, the outcome can be disastrous for the owner.

What is the fair price for the right to buy or sell a certain derivative for some predetermined exercise dates? And can the existing classical methods be improved by using deep learning? The thesis tries to answer these questions through analytical and numerical methods in the Black-Scholes model. The aim is to give an accurate and fast method for arbitrage free pricing with neural networks, where classical methods will be used for comparison. The focus in the thesis will be on financial equity contingent claims, where the prime example will be American equity put options with one or two underlying stocks.

This thesis starts presenting the basic theory on arbitrage theory to introduce the basic modeling framework for contingent claims. With this introduction, the framework is established to explore the applications and numerical procedures that spring from the underlying theory. The classical methods in chapter 3, such as the binomial lattice approach and least squares Monte Carlo methods (LSM) will be applied to univariate and bivariate contingent claims. The methods provide a benchmark for

the neural network approaches for option pricing in later chapters. Especially the lattice approach gives strong intuition on how the optimal stopping problem can be solved, where the LSM provides a framework that is easily extended to methods for pricing multivariate contingent claims. The lattice approach makes it easy to compare values with closed form solution for European options, where some special cases for exotic European options will be presented.

Deep learning theory is important for building a sound and high quality model for pricing options with neural networks. Hence, the basic machine learning and deep learning theory will be presented before the main chapter. Chapter 5 is the main chapter, where we present pricing methods for contingent claims with neural networks. The aim is to look for methods that have high accuracy and fast computation time. Chapter 6 gather all the methods presented and compare them numerically. Chapter 6 also contains a discussion on the pricing methods and the underlying Black-Scholes theory. Finally, in chapter 7, we conclude on our findings and suggest what can be further investigated.

Chapter 2

Arbitrage Theory in Continuous Time Finance

Arbitrage theory in continuous time finance is a field with a lot of technical details from probability theory and stochastic calculus, where we follow the style in (Hull, 2017; Björk, 2009) to focus on intuition without going into the whelm of technicalities and proofs. The focus of this chapter is to provide the basic tools and lay the foundations for the pricing methods. The key question is how to price derivatives fairly and hedge the risk imposed by them. The thesis will mainly deal with the former, where the concepts of arbitrage and replication will be important.

We begin by introducing the financial markets and key concepts for building a arbitrage-free and complete market model (section 2.1). Then we build a framework for finding the "fair" price, i.e. finding a arbitrage-free and complete model (section 2.2). Lastly, we go into specific cases where either a closed form solution exists or numerical methods are needed (section 2.3 and 2.4).

2.1 Financial Markets

In the financial markets, there are many agents and different types of investments. The classical investment types are bonds and stocks, where the big players in the markets are commercial banks, investment banks, insurance companies, and pension funds. Derivatives provide additional options for investment.

A derivative or a contingent claim is a financial instrument depending on an underlying asset(s), where the dependency is specified in the contract. We will focus on contingent claims with one or two underlying stocks, i.e. univariate and bivariate contingent claims, but the techniques developed can easily be extended to other types of derivatives.

In a general setting, to find prices of contingent claims we restrict our financial market to d risky assets $\boldsymbol{S}(t) = (S_1(t), S_2(t), \ldots, S_d(t))$ and a bank account $S_0(t)$. The probability space $(\Omega, \mathcal{F}, P)$ with a filtration $\mathbb{F} = (\mathcal{F}_t)_{t\geq 0}$ is fundamental for modeling stochastic processes describing asset prices and trading strategies, where, in the thesis, the filtered probability space $(\Omega, \mathcal{F}, \mathbb{F}, P)$ will be implicitly assumed. Intuitively the filtration $\mathcal{F}_t$ is the information observable at time t, where the filtration $\mathbb{F}^W$ generated only by the Wiener processes $(W_t)_{0\leq t\leq T}$ will be important for having a complete market.

The bank account is assumed to be a strictly positive adapted process $S_0 = (S_0(t))_{t\geq 0}$ and $S_0(0) = 1$, where the d risky assets are modeled by a $\mathbb{R}^d$ adapted stochastic process $\boldsymbol{S} = (\boldsymbol{S}(t))_{t\geq 0}$. The risky assets are stocks where the stocks are assumed positive $S_i(t) \geq 0$ P-a.s for all i and $t \geq 0$ for financial reasons. By using the bank account as numéraire, i.e. dividing the traded asset by the bank account $(\frac{S(t)}{S_0(t)})$, amounts to working with *zero interest*. We assume that our financial market is frictionless.

Assumption 2.1. ***Frictionless Market:*** *We assume the following institutional facts:*

- *Short positions and fractional holdings are allowed*
- *There is no bid-ask spread, i.e. selling price is equal to buying price*
- *There are no transactions costs, taxes, or margin requirements of trading*
- *The market is completely liquid, i.e. it is possible to buy/sell unlimited quantities on the market. You can borrow an unlimited amount from the bank by short selling*

(p. 6 (Björk, 2009))

Besides the assumptions in (Björk, 2009), we assume the market gives the same uniform price for borrowing money. Stocks are fixed stochastic processes exogenously and a priori given. All the assumptions do not reflect real market conditions, but they are mathematically convenient.

2.1.1 Contingent Claims

A contingent claim is a contract on an underlying asset or assets, where the price of the claim is contingent on the price behavior of the underlying asset(s). A bivariate contingent claim refers to the option being dependent on two risky assets[1]. We investigate stock derivatives with different types of contracts, where we will mainly divide the derivatives into two classes.

1. Simple European derivatives
2. Exotic derivatives (e.g. American options)

Simple European options can only be exercised at maturity (time T) and they depend only on one underlying asset. We have a closed form solution for the simple European options (section 2.16). The exotic derivatives are a broad class of functions on the underlying asset(s), where you can e.g. have an American option where the holder can exercise from inception to maturity (section 2.4) or a contract on several underlying stocks. Examples of simple European derivatives are the European call and put options.

Definition 2.2. European Call and Put Options: A European call option is an option where the owner has the right to buy the underlying asset to price K at maturity. If the owner of the option chooses to buy the underlying asset, then the option is exercised. The contract function for the European call option is:

$$\Phi(S(T)) = \max\{S(T) - K, 0\}$$

[1]Similar refers the multivariate contingent claim to that the claim depends on two or more underlying assets

The put option is the right to sell the underlying asset to price K at maturity, hence the contract function for the European put option is:

$$\Phi(S(T)) = \max\{K - S(T), 0\}$$

Where S(T) is the price of the underlying asset at maturity and K is the agreed strike price.

The American option adds the feature to the European option, that you can exercise at anytime between the inception of the contract until maturity (section 2.4). The payoff function is the same for the European and American option. The difference between the European and American option lies in the exercise opportunities. Figure 2.1 illustrates the European call and put options payoff at maturity.

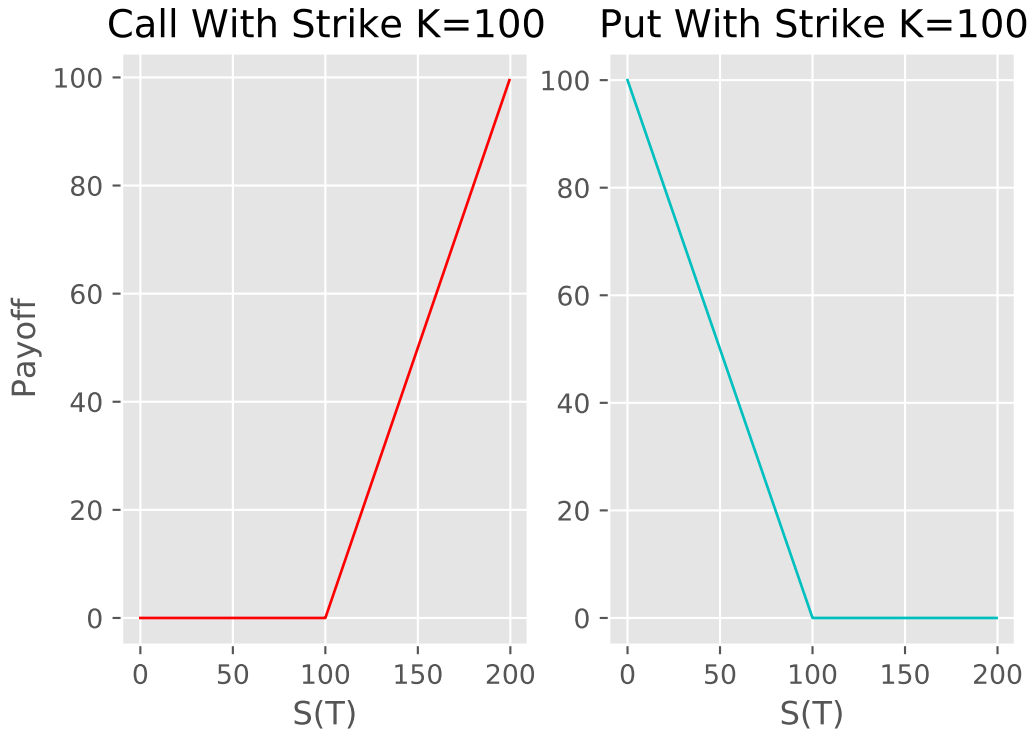


FIGURE 2.1: European options payoff at maturity with strike K=100

Figure 2.1 shows that the owner of the option has a limited downside, but the graph does not take into account the initial price for the option. The profit and loss ($P\&L$) graph is also a common way of illustrating the payoff for an option, where the initial cost of buying the option is taking into account. The European call and American put will be the central derivatives considered together with an American bivariate contingent claim. The task ahead is to find the initial price or the fair price for these contracts, where the concepts of completeness and arbitrage will be central.

2.1.2 Self-financing Portfolio (Without Consumption)

Before being able to use the concepts of arbitrage and completeness, the construction of the portfolio from the financial market model will be important. The portfolio is

the number of each asset from the market the owner of the portfolio holds. The value of the portfolio for a market model with the bank account and d stocks is:

$$V^h(t) = \sum_{i=0}^{d} h_i(t)S_i(t)$$

V^h is called the value process and $h_i(t)$ is the number of shares of type i during the period "$[t, t+dt)$". For the definition of arbitrage (Definition 2.5) we need to restrict ourselves to self-financing (S-F) portfolios. A self-financing portfolio h, is a portfolio h which does not get any external injection of money.

Definition 2.3. Self-financing portfolio: A portfolio consisting of d+1 asset(s): h(t)=$(h_0(t), h_1(t), \ldots, h_d)$ is self-financing if:

$$dV^h(t) = \sum_{i=0}^{d} h_i(t)dS_i(t)$$

Where S_i is the $i'th$ asset in our portfolio.

(p. 87 (Björk, 2009))

When dealing with discrete time finance the S-F portfolio is a budget restriction. This is important intuition for the continuous time version because the continuous time version can be thought of as the limit of the discrete version by letting step sizes in time tending to zero. To avoid pathological effects on the portfolio one often introduces the concept of an admissible portfolio:

Definition 2.4. a-admissible portfolio: For some $a \geq 0$, a portfolio h is called a-admissible if its value process $V^h(t)$ is uniformly bounded from below by -a.

(p. 139 (Björk, 2009))

The definition of a-admissible portfolio is to avoid situations as the doubling strategy known from gambling which imposes a limit to the debt arrangement. The important takeaway is that the S-F portfolio is a portfolio where you only reallocate your assets through time within the portfolio.

2.1.3 Arbitrage

Arbitrage is the financial term for a *"free lunch"*. An arbitrage opportunity produces something out of nothing without risk. For an efficient and well-functioning market the *"money pumps"* cannot exist for long because the *"free lunch"* would quickly be eroded by exploitation. To avoid making a *"money machine"* in our market, we want to price derivatives by not introducing arbitrage to the market.

Definition 2.5. Arbitrage: An arbitrage possibility on a financial market is an admissible self-financed portfolio h such that

$$\begin{aligned} V^h(0) &= 0 \\ P(V^h(T) \geq 0) &= 1 \\ P(V^h(T) > 0) &> 1 \end{aligned}$$

The financial market $\boldsymbol{S}$ is called arbitrage-free if there exists no arbitrage opportunities.

(p. 96 (Björk, 2009))

From the above definition we see that arbitrage is a natural financial requirement for a financial market model because the investor in an arbitrage portfolio can start with 0 dollars, and without injecting any money, the investor is certain of not losing any money. In addition, the investor has a positive probability of ending up with more than 0 at maturity. This cannot be a well-functioning market for both buyers and sellers. To price the derivatives fair in the model, the derivative should not introduce arbitrage to the market. An arbitrage-free market is not the only desirable property for the market, we would also like to have a unique price for the derivatives. We can obtain a unique price by replication of the cash flow from the derivative with the other assets in the market model. If every derivative can be replicated the market is complete.

2.1.4 Complete Market and Replication

The replication argument in Black-Scholes paper (Black and Scholes, 1973) was groundbreaking in the sense that the attitude to risk was irrelevant for pricing because by continuous trading in the underlying asset(s) the cash flow from the contingent claim could be replicated. The replication argument shows that the price is unique under the assumption that investors prefer more to less.

Replication is also important for risk management of the derivative books because it tells you how to risk neutralizes your exposure[2]. In the definition below, we define a replication for a simple T-claim[3].

Definition 2.6. Replication and completeness for T-claim: A T-claim X can be replicated, if there exists a self-financing portfolio h such that:

- $V^h(T) = X$ P-a.s.

I.e. h is a replication portfolio for X if it is guaranteed to pay in all circumstances an amount identical to the payout of the derivative X. The market is complete if every derivative in the market can be replicated.

(p. 192 (Björk, 2009))

By introducing the basic concepts for how to price fairly and protect ourselves against financial risk, we will in the next section focus on building the financial market model.

2.2 Multidimensional Models

There are two main methods for building a arbitrage-free and complete market model. The classical approach is the delta hedging approach (Black and Scholes, 1973) and (Cox and Stephen Ross, 1979)). The more advanced mathematical approach is the martingale approach (Björk, 2009). In this section, we focus on the martingale approach and show that the delta hedging approach coincides with the more general martingale theory. For the martingale approach, the First and Second Fundamental Theorems of Mathematical Finance will be the key to obtain a fair market. Besides the financial market assumptions in section 2.1 will we assume specific model assumptions.

[2]Note there is a subtle difference between to hegde or replicate a cash flow. The hedge gives minus the cash flow from replication

[3]Options only exercisable at maturity

2.2.1 Model Assumptions

Let us consider a filtered probability space $(\Omega, \mathcal{F}, P, (\mathcal{F}_t^{\bar{W}})_{t\in[0,T]})$. Note the assumption that the filtration is generated from the Wiener process and we consider a finite horizon. We assume $\bar{W}_i$ is k-dimensional and $\bar{W}$ is the only random source. A priori we assume a market $(B(t), S_1(t), S_2(t), \ldots, S_d(t))$, where $S_i(t)_{i=1,2,\ldots,d}$ are d risky assets and B(t) is the risk-free asset[4]. We assume the bank account and risky dynamics are given by:

$$d\boldsymbol{S}(t) = D[\boldsymbol{S}(t)]\boldsymbol{\alpha}(t)dt + D[\boldsymbol{S}(t)]\boldsymbol{\sigma}(t)d\bar{\boldsymbol{W}}(t) \qquad S_i(0) \in \mathbb{R}^+ \qquad (2.1)$$

$$dB(t) = r(t)B(t)dt \qquad B(0) = 1 \qquad (2.2)$$

We assume $\alpha_i(t)$, $\sigma_{ij}(t)$ and the short rate $r(t)$ are adapted and bounded processes, these conditions are necessary for the stochastic integrals to be well-defined. The evolution of the stocks are described by the geometric Brownian motion (GBM) which has a solution to the SDE. The randomness comes from the Wiener process[5] in the GBM, which has wild trajectories. The function $t \mapsto W(t,\omega)$ from $[0,\infty)$ to $\mathbb{R}$ is continuous but nowhere differentiable. Furthermore, the Wiener process has nonzero quadratic variation and infinite variation, which is the reason for stochastic calculus pioneered by Itô. The Wiener process also has well-behaved property, e.g. it is a Lévy process:

- $W(0) = 0 \quad a.s$
- Independent and stationary increments $W(t) - W(s)$ which is normally distributed with mean zero and variance t-s
- W has continuous trajectories

(p. 40 (Björk, 2009))

The Wiener process in the GBM formalizes *"random shocks"* dW to the stock return with volatility $\boldsymbol{\sigma}(t)$ and drift $\boldsymbol{\alpha}(t)$. Figure 2.2 illustrates three approximations to sample paths of the stocks with GBM assumption and initial value $S_i(0) = 36$.

The tool for handling Wiener processes is stochastic calculus in continuous time because the standard calculus will not work for the wildly behaved Wiener process. In the representation of the GBM, we used vector and matrix notation. The stock vector is d dimensional and the Wiener process vector is k dimensional. The volatility matrix is given by $\boldsymbol{\sigma}(t) = \{\sigma_{ij}(t)\}_{i=1,\ldots,d,j=1,\ldots,k}$ and the drift is $\boldsymbol{\alpha}(t) = (\alpha_1(t), \alpha_2(t), \ldots, \alpha_d(t))^T$. D(x) denotes a diagonal matrix with vector x as its diagonal and the Wiener processes instantaneous correlation matrix $\boldsymbol{\rho}$ is given by $Cov(dW_i(t), dW_j(t)) = E(dW_i(t)dW_j(t)) = \rho_{ij}dt$.

2.2.2 Arbitrage-free Model

The first problem we are faced with in arbitrage theory is to create a model with no arbitrage opportunities. The First Fundamental Theorem tells us how to not introduce arbitrage to our market model.

[4] B(t) is often called the bank account

[5] A Wiener process is also called a Brownian motion

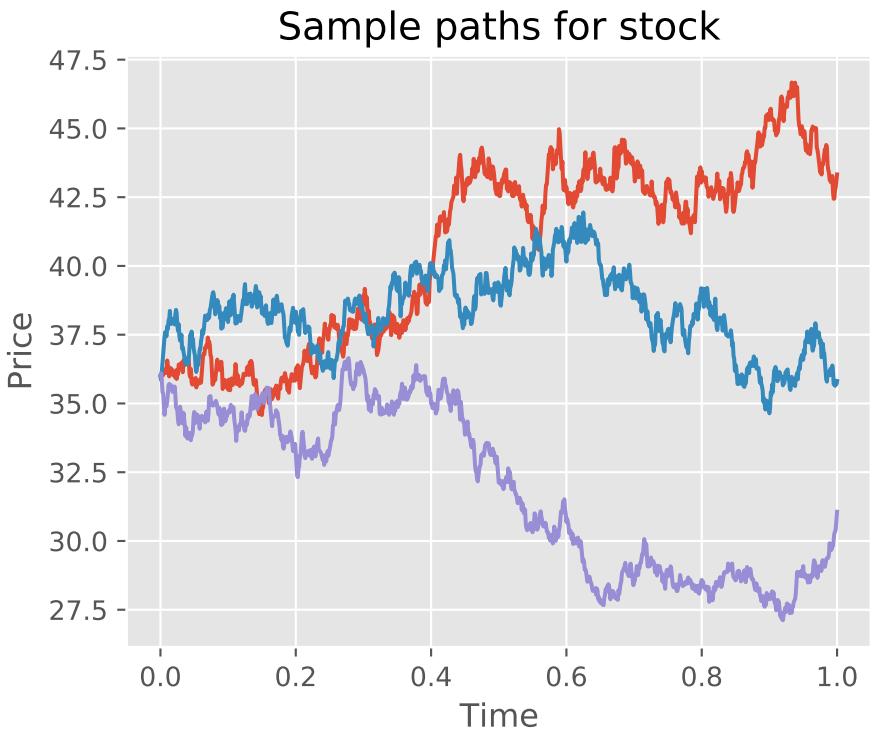


FIGURE 2.2: Three sample paths for a stock under GBM assumption, where the spot is \$36, σ=0.2, and α=0.06

Theorem 2.7. ***First Fundamental Pricing Theorem of Mathematical Finance(FFT1):*** *The market model is free of arbitrage if and only if there exists an equivalent martingale measure, i.e. a measure $Q \sim P$ such that the processes:*

$$\frac{S_0(t)}{S_0(t)}, \frac{S_1(t)}{S_0(t)}, \cdots, \frac{S_d(t)}{S_0(t)}$$

are (local)martingales under Q.

(p. 154 (Björk, 2009))

Assuming $S_0(t)$ is the bank account $B(t)$, then the martingale property for the discounted price processes is a mathematical formulation that the expected discounted value coincides with the known spot value today[6]. In gambling, a martingale resembles a *"fair"* game. From the FFT1 using the bank account $B(t)$ as numéraire, it follows that:

Proposition 2.8. *We assume that $B(t) = S_0(t)$ is our numéraire and all the processes' randomness comes from the Weiner process, then an equivalent measure $Q \sim P$ is a martingale measure if and only if all assets $(B(t), S_1(t), \ldots, S_d(t))$ have the short rate as their local rates of return, i.e.*

$$dS_i(t) = S_i(t)r(t)dt + S_i(t)\sigma_i(t)d\boldsymbol{W}(t)$$

(p. 154 (Björk, 2009))

[6] Assuming $B(0) = 1$

So, to not introduce arbitrage to the model for the financial market, we need to ensure the Q-dynamics of S is:

$$d\boldsymbol{S}(t) = D[\boldsymbol{S}(t)]\boldsymbol{r}(t)dt + D[\boldsymbol{S}(t)]\boldsymbol{\sigma}(t)d\boldsymbol{W}(t) \tag{2.3}$$

The tool to obtain the dynamics in equation (2.3) is the Girsanov theorem (theorem A.3). Girsanov theorem is a continuous measure transformation, where, in our model, we want to transform the dynamics given with the objective probability measure P to an equivalent martingale measure Q. By suitable choices of the likelihood process L and setting $dQ = L(T)dP$, then with Girsanov theorem the transformed process is still a Brownian motion:

$$d\bar{\boldsymbol{W}}(t) = \boldsymbol{\phi}(t)dt + d\boldsymbol{W}(t)$$

When applying to eq. (2.1):

$$d\boldsymbol{S}(t) = D[\boldsymbol{S}(t)](\boldsymbol{\alpha}(t) + \boldsymbol{\sigma}(t)\boldsymbol{\phi}(t))dt + D[\boldsymbol{S}(t)]\boldsymbol{\sigma}(t)d\boldsymbol{W}(t)$$

Going back to the FFT1 and the proposition hereof, we know that Q is a martingale measure if and only if:

$$\boldsymbol{\alpha}(t) + \boldsymbol{\sigma}(t)\boldsymbol{\phi}(t) = \mathbf{r}(t) \quad \textit{holds with probability } 1 \textit{ for each } t \tag{2.4}$$

We disregard pathological models, when doing so, the term *"generically arbitrage-free"* will be used.

Definition 2.9. Generically arbitrage-free: The model in this section is said to be generically arbitrage-free if it is arbitrage-free for every (sufficiently integrable) choice of $\boldsymbol{\alpha}(t)$.

(p. 198 (Björk, 2009))

Furthermore, we assume enough integrability and we have the following useful result:

Proposition 2.10. *Disregarding integrability problems the model is generically arbitrage-free if and only if, for each $t \leq T$ and P-a.s. the mapping: $\boldsymbol{\sigma}(t) : \mathbb{R}^k \to \mathbb{R}^d$ is surjective, i.e. if and only if the volatility matrix $\boldsymbol{\sigma}(t)$ has rank d.*

(p. 198 (Björk, 2009))

We note that to have an arbitrage-free model, we need $k \geq d$, i.e. have at least as many random sources as the number of risky assets.

2.2.3 Complete model

Second Fundamental Pricing Theorem is key to obtain a complete market model, i.e. a market model with unique prices where every claim can be hedged.

Theorem 2.11. ***Second Fundamental Pricing Theorem of Mathematical Finance(FFT2):*** *Assuming absence of arbitrage, the market model is complete if and only if the martingale measure Q is unique.*

(p. 155 (Björk, 2009))

In our Wiener world, we have a unique martingale measure if equation 2.4 has a unique solution. The proof of proposition 2.12 will shortly reveal why the Wiener

world assumption is required. If we had more random sources e.g. a Poisson process[7], then there is no guarantee that the equivalent measure transformation is of the Girsanov type above.

Proposition 2.12. *Assume that the model is generically arbitrage-free and that the filtration is defined by:*

$$\mathcal{F}_t = \mathcal{F}_t^{\bar{W}} \quad t \in [0, T]$$

Then disregarding integrability problems, the model is complete if and only if k=d and the volatility matrix $\sigma(t)$ is invertible P-a.s. for each $t \leq T$

Proof. The proof is based on martingale representation theorem A.2 (MRT) and converse of Girsanov theorem A.4 which uses MRT, hence the assumption about the only randomness comes from the Wiener process. By the two theorems, we know every equivalent measure transformation is obtained by the Girsanov theorem of the above type. Hence, the martingale measure is unique if and only if the solution to (2.4) is unique. ■

(p. 200 (Björk, 2009))

Intuitively we need one independent traded asset excluding the bank account for every source of randomness (Meta-theorem 8.3.1 (Björk, 2009)). Henceforth we will not distinguish between arbitrage-free and generically arbitrage-free.

2.2.4 Pricing and Connection to Classical Approach

The pricing formula for arbitrage-free market model is the risk neutral valuation formula(RVNF):

Proposition 2.13. ***Risk Neutral Valuation Formula:*** *To avoid arbitrage, the derivative $\mathcal{X}$ must be priced according to the formula:*

$$\Pi(t; \mathcal{X}) = S_0(t) E^Q[\frac{\mathcal{X}}{S_0(T)} | \mathcal{F}_t] \tag{2.5}$$

Note if we choose our numéraire $S_0(t) = B(t)$ then

$$\Pi(t; \mathcal{X}) = E^Q[\exp(-\int_t^T r(s)ds)\mathcal{X} | \mathcal{F}_t] \tag{2.6}$$

(p. 155 (Björk, 2009))

Proposition 2.13 will raise the question if there is more than one fair price for the derivative. The answer is found in FTT2, the market is complete if and only if the measure Q is unique. The conditional expectation in the RNVF is a natural choice for pricing, because intuitively, the conditional expectation is our best estimate given the available information[8].

The classical approach in (Black and Scholes, 1973) to a arbitrage-free and complete market model is based on a Markovian model assumption. For the model to have the Markov property, we assume $k = d$ and the probability space is $(\Omega, \mathcal{F}, P, \mathcal{F}_t^{\bar{W}_t})$.

[7] The Merton's Mixed Jump-Diffusion Model is an example

[8] Mathematically known as the projection property

Furthermore, we assume $\boldsymbol{\alpha}$ and σ are deterministic and constant over time. σ is also assumed invertible. Under these more restrictive assumptions, the risk neutral valuation formula for a simple T-claim is given by the pricing function:

$$F(t, \boldsymbol{S}(t)) = \exp(-r(T-t))E^{Q}[\mathcal{X}|\boldsymbol{S}(t)] \tag{2.7}$$

The Markov property implies that the price only depends on the current state of $\boldsymbol{S}$. Applying Kolmogorov backward equation on equation (2.7) we obtain the Black-Scholes PDE for the pricing function $F(t, \boldsymbol{S}(t)) = \Pi(t; \mathcal{X})$.

Theorem 2.14. ***Black Scholes PDE:*** *Consider the contract $\mathcal{X} = \Phi(\boldsymbol{S}(T))$. In order not to introduce arbitrage to the market, the pricing function $F(t,s)$ must solve the boundary value problem.*

$$F_t(t,s) + \sum_{i=1}^{d} rs_i \frac{\partial F_i(t,s)}{\partial s_i} + \frac{1}{2}\sum_{i,j=1}^{d} s_i s_j \frac{\partial F(t,s)}{\partial s_i \partial s_j}(\sigma\sigma^T)_{i,j} - rF(t,s) = 0$$

$$F(T,s) = \Phi(s)$$

(p. 203 (Björk, 2009))

2.3 Classical Black-Scholes Theory

We will not do the classical delta hedging approach in (Black and Scholes, 1973). Instead, we use the general multidimensional martingale approach to derive the essential formulas for pricing. To derive a closed form solution to the European call and put option, we concentrate on a special case of the multidimensional framework, where we only have the risk-free asset and one risky asset in the financial market model. We further restrict ourselves to:

Assumption 2.15. ***Black-Scholes assumptions****: We assume the following ideal conditions in addition to (2.1):*

- *The short-term interest rate $r \in \mathbb{R}_*^+$[9], volatility $\sigma \in \mathbb{R}^+$ and the drift $\alpha \in \mathbb{R}$ are constant.*
- *The stock pays no dividends or other distributions.*
- *The option is simple ("European").*
- *No arbitrage opportunity on the market.*

(p. 640 (Black and Scholes, 1973))

The interest rate is assumed strictly positive to assure the European call and American call value coincides[10]. The above assumptions gives the Markovian model described in the previous section.

[9] Note that restricting the interest rate to the positive reals is not part of the Black-Scholes papers assumptions.

[10] Detailed explained in section 2.4.1

We assume the underlying stock and the bank account have differentials:

$$\begin{aligned} dS(t) &= S(t) \cdot \alpha dt + S(t)\sigma d\bar{W}(t) & S(0) &\in \mathbb{R}^+ \\ dB(t) &= rB(t)dt & B(0) &= 1 \end{aligned}$$

By Itô's lemma (lemma A.1) for one dimensional process the solution to the differentials above is:

$$\begin{aligned} S(t) &= S(0) \cdot \exp\left((\alpha - \frac{1}{2}\sigma^2)t + \sigma W(t)\right) \\ B(t) &= \exp(r \cdot t) \end{aligned}$$

The solution of the SDE of S under Q dynamics is:

$$S(t) = S(0) \cdot \exp\left((r - \frac{1}{2}\sigma^2)t + \sigma W(t)\right) \tag{2.8}$$

By equation (2.8) we see that the Black-Scholes model assumes that the stock price evolution produces a lognormal distribution for the price at any future time.

The closed form solution for the European call can be derived from solving the Black-Scholes PDE or with the RNVF given in previous section.

Proposition 2.16. ***Black-Scholes formula for a call option:*** *The price of a European call option with strike K and maturity T is given by the formula* $\Pi(t) = F(t, S(t)$*, where*

$$F(t,s) = c(t,s) = s \cdot N(d_1(t,s)) - \exp(-r(T-t)) \cdot K \cdot N(d_2(t,s))$$

N is the cumulative distribution function of a standard normal distribution $\mathcal{N}(0,1)$ *and*

$$\begin{aligned} d_1(t,s) = \frac{1}{\sigma \cdot \sqrt{T-t}} \cdot \left(\ln(\frac{s}{K}) + (r + \frac{1}{2}\sigma^2)(T-t)\right) \\ d_2(t,s) = d_1(s,t) - \sigma\sqrt{T-t} \end{aligned}$$

(p. 105 (Björk, 2009))

We provided only the price for the European call option, but the European put price can readily be obtained by the put-call-parity for European options.

Proposition 2.17. ***Put-call parity:*** *Assume the call and put option has the same strike price and time to maturity.*

$$p(t,s) = K \cdot \exp(-r(T-t)) + c(t,s) - s$$

(p. 126 (Björk, 2009))

The aim of this thesis is to price American put options, but the European option provides a reference price in a closed form. The put-call-parity holds only for European options, where for the American option there is a bound on price difference:

$$S_0 - K \leq C - P \leq S_0 - K \cdot \exp(-rT)$$

The above formula for the European call option is the same for an American call option, but is not true for an American put option or call options with underlying

stock paying dividends. The result for the American call option was shown by Merton (Merton, 1973), that the intrinsic value is never greater than the worth of the option given by the risk-neutral valuation formula. In section 2.4 we will show a martingale approach to prove the value of a European and American call coincides when the underlying is a non-dividend paying stock.

2.4 American Options and Optimal Stopping

The American options add additional complexity to the pricing problem, because compared to the European option the American option can be exercised at anytime from inception to maturity. The exercise value at time t is also called the intrinsic value of the option. This section is inspired by (Björk, 2009; Shiryaev and Peskir, 2006; Elliott and Kopp, 1999) where (Shiryaev and Peskir, 2006) is specialized to optimal stopping problems and the two other references provide the fundamentals for option and arbitrage theory in general.

We still assume a diffusion setting that the underlying stochastic process for the stock behaves under the risk neutral measure as a GBM. The exercise feature of the American option raises the problem of rationally finding the optimal stopping time to maximize profit. We will assume a finite horizon $T \in \mathbb{R}_*^+$ throughout the thesis, because all the derivatives will be priced in a finite timeframe. Let the gain function $G : \mathbb{R} \to \mathbb{R}$ be a measurable function satisfying:

$$E_x[\sup_{0\leq t\leq T} |G(S(t))|] < \infty \tag{2.9}$$

where S is the underlying stochastic process. If the integrability condition is satisfied on a finite interval $[0, T]$ (equation (2.9)) then the optimal stopping problem for gain function G and $x \in \mathbb{R}$ is well defined. We assumed that the underlying stochastic S(t) process is time-homogeneous, but the assumption can be relaxed. If S(t) is a time-inhomogenous we can extend the underlying process S(t) by time, albeit increasing the underlying process dimension[11]. We define the optimal value process in terms of the gain process.

Definition 2.18. For fixed $(t, x) \in [0, T] \times \mathbb{R}$, and each stopping time τ with $\tau \geq t$ the optimal value function $V(t, x)$ is defined by

$$V(t, x) = \sup_{\tau\in\mathcal{T}^T} E_{t,x}[G(S(\tau))] \tag{2.10}$$

A stopping time that realizes supremum for V is called optimal and denoted $\hat{\tau}$.
(p. 341 (Björk, 2009))

The solution to the optimal stopping problem $\hat{\tau}$ is where supremum is attained and the price is then $V(t, x)$ for $(t, x) \in [0, T] \times \mathbb{R}$. The supremum is taken over all stopping times with respect to the natural filtration $\mathcal{F}_t$ belonging in the class of stopping times:

$$\mathcal{T}_0^T = \mathcal{T}^T = \{\tau : 0 \leq \tau \leq T\}$$

The definition of a stopping time τ can be seen in definition A.5. The intuition is that the stopping time is a random time, where we know, at the present time, whether

[11] This trick is seen in life insurance for models with state duration, where you include the duration process

the process is stopped or not. To solve the optimal stopping problem some trivial solutions are immediate by martingale properties:

Proposition 2.19. *The following holds:*

- *If $G(S(t))$ is a submartingale, then it is not optimal to stop at all and $\tau^* = T$*
- *If $G(S(t))$ is a martingale, then all stopping times $\tau \in [0, T]$ are optimal*
- *If $G(S(t))$ is a supermartingale, then it is optimal to stop immediately. i.e. $\tau^* = 0$*

(p. 330 (Björk, 2009))

Examples of optimal stopping problems could be the American call and put options:

$$C(t,x) = \sup_{\tau \in \mathcal{T}_t^T} E^Q[\exp(-r(\tau - t))(S(\tau) - K)^+ | S(t) = x] \quad \text{for } t \in [0,T] \text{ and } x \in \mathbb{R}^+$$
$$P(t,x) = \sup_{\tau \in \mathcal{T}_t^T} E^Q[\exp(-r(\tau - t))(K - S(\tau))^+ | S(t) = x] \quad \text{for } t \in [0,T] \text{ and } x \in \mathbb{R}^+$$

2.4.1 American Call without Dividends

The American call option is a special case, because the optimal stopping time is always at the maturity of the option. With martingale machinery, this means the gain function is a submartingale which implies that $\hat{\tau} = T$ (proposition 2.19). Remember the optimal stopping problem for an American call option:

$$C(t,x) = \sup_{\tau \in \mathcal{T}_0^{T-t}} E_{t,x}^Q[\exp(-r\tau)(S(t+\tau) - K)^+]$$

Looking at the gain function:

$$\exp(-rt)(S(t) - K)^+ = (\exp(-rt)S_t - \exp(-rt)K)^+$$

Recall that the discounted price process $\exp(-r \cdot t) \cdot S_t$ is a Q-martingale and $\exp(-r \cdot t) \cdot K$ is a deterministic decreasing function in t if $r > 0$. Furthermore, the function $x \mapsto (x)^+$ is convex and increasing, hence the gain function is a Q-submartingale. The last result used is that a convex and increasing function on a submartingale is still a submartingale, hence the optimal stopping time is $\hat{\tau} = T$ if $r > 0$.

2.4.2 American Put

The arbitrage-free price for an American put at time t:

$$P(x,t) = \sup_{\tau \in \mathcal{T}_t^T} E^Q[\exp(-r(\tau - t))(K - S(\tau))^+ | S(t) = x] \quad \text{for } t \in [0,T] \text{ and } x \in \mathbb{R}^+ \tag{2.11}$$

For the American put, we need computational methods because the American and European put do not coincide as for the call option.

Proposition 2.20. *Consider the European and American put option with same maturity T and strike K. If the risk-free rate $r \in \mathbb{R}_*^+$, then for any $t < T$*

$$p(x,t) < P(x,t) \tag{2.12}$$

Proof. WLOG[12] we assume that t=0. Define the stopping time

$$\tau = min\{t \geq 0 : S(t) \leq K\Big(1 - \exp\Big(-r \cdot (T-t)\Big)\Big)\}$$

We consider an exercise strategy $min\{\tau, T\}$, where the strategy is not necessarily optimal. We consider two events:

1) $\tau < T$

2) $\tau \geq T$

The first case is to exercise at time τ when $S(\tau) \leq K(1 - \exp(-r \cdot (T - \tau)))$. Here the payoff by exercising will be at least $K \exp(-r \cdot (T - \tau))$. The cash flow received is then invested into the bank account at time τ. At maturity, the strategy gives the holder of the put a payoff K, where the European contract with strike K will pay less, because the stock price at maturity will be $S(T) > 0$ *a.s.*

The second case is trivial, because the European and American put will give the same payoff. Since the first case has a positive probability, the American put has a higher discounted expected payoff by following the above strategy regardless of the spot price for the stock. ■

The above proposition shows that the optimal stopping strategy is not always to hold the option to maturity, hence the theory of optimal stopping is important for the pricing of American put options. The American put has consequently no closed form solution, but we can search for a lower exercise boundary b(t) such that the holder of the option should exercise when:

$$S(\tau) \leq b(\tau) \quad \tau \leq T$$

The continuation set C and stopping set $\bar{D}$ are given for an American put.

$$C = \{(t,x) \in [0,T) \times (0,\infty) : P(t,x) > G(x)\}$$
$$\bar{D} = \{(t,x) \in [0,T) \times (0,\infty) : P(t,x) = G(x)\}$$

Hence the first optimal stopping time after time t for the American put is

$$\hat{\tau} = \inf\{u \in [t,T] : P(S(u),u) = (K - S(u))^+\}$$

2.4.3 Discrete Time Valuation

To solve the optimal stopping problem numerical methods are required for the American put option. The computer is discrete by nature, hence we choose to either approximate the Black-Scholes model with a discrete time model or only consider discrete time exercise opportunities. The first approach is the lattice approach, where the binomial model describes the evolution of the stock. The second approach is to approximate the American put option by the Bermudan option. Either way, we will consider a discrete time probability space and the discrete time optimal stoppping problem.

[12] Without loss of generality

Suppose the probability space $(\Omega, \mathcal{F}, P)$ is equipped with the natural discrete filtration $(\mathcal{F}_{t_n})_{n=0,1,\ldots,N}$ modeling a financial market. The tenor structure[13] is that the time to maturity is divided into a grid of N+1 equidistant points in time $0 = t_0 \leq t_1 \leq t_2, \cdots \leq t_N = T$, where $\Delta t_n = t_n - t_{n-1} = T/N$ for each $n = 1, \ldots, N$. At each grid point or exercise date the option holder chooses to exercise or keep the option alive.

The underlying process is assumed to be Markovian with state variables $(S(t_n))_{n=0,1,\ldots,N}$ recording all necessary information about relevant financial variables adapted to the natural filtration in order to solve the optimal stopping problem with the dynamic programming principle. Furthermore, the gain process is adapted to the filtration and square integrable $E[\max_{0\leq n\leq N} |G(S(t_n))|^2] < \infty$, which is a necessary condition for using regression (section 3.3).

The optimal stopping problem in discrete time is to find:

$$\sup_{\tau \in \mathcal{T}(0,1,\ldots,T)} E^Q[G(S(\tau))] \tag{2.13}$$

For using the programmable dynamic programming principle the Snell envelope $(U(t_n))_{n=0,1,\ldots,N}$ (definition A.6) of the gain function $G(S(t_n))_{n=0,\ldots,N}$ is useful and defined by:

$$U(t_n) = \operatorname*{ess\,sup}_{\tau \in \mathcal{T}(n,\ldots,N)} E^Q[G(S(\tau))|S(t_n)] \quad n = 0, 1, \ldots, N$$

The Snell envelope $U(t_n)$ is the smallest supermartingale of the gain process $\{G(S(t_n))\}$, i.e. the smallest supermartingale dominating the gain process. Using the Snell envelope theory the optimal stopping problem can be solved with the dynamic programming principle.

$$\begin{cases} U(t_N) = G(S(t_N)) \\ U(t_n) = max\{G(S(t_n)), E^Q[U(t_{n+1})|S(t_n)])\} \quad for\ n = 0, \ldots, N-1 \end{cases} \tag{2.14}$$

Where $U(t_n)$ is the discounted option value at time t_n not previously exercised and $G(S(t_n))$ is the discounted exercise value.

Equation (2.14) is known as value iteration and indicates that the holder should exercise the first time $G(S(t_n)) > E^Q[U(t_{n+1})|S(t_n)]$ to maximize the profit from the option. Therefore, the strategy gives the first optimal stopping time. Note that an optimal stopping time will always exist in discrete time when $T < \infty$, but there is no guarantee that the stopping time is unique. The value iteration provides the optimal value process of the gain process $G(S(t))$ (theorem A.7). So

$$U(t_n) = E^Q[G(S(\tau_{t_n})|S(t_n)]$$

with

$$\tau_{t_n} = \min\{k \geq n : U(t_k) = G(S(t_k))\}$$

and

$$E^Q[U(0)] = \sup_{\tau \in \mathcal{T}(0,\ldots,T)} E^Q[G(S(\tau))] = E^Q[G(S(\hat{\tau}))]$$

[13] The tenor of an option is the remaining life of the option from today to maturity

The optimal stopping problem can also be solved in terms of stopping times instead of using the value process. This alternative dynamic programming recursive equation is called policy iteration.

$$\begin{cases} \tau_{t_N} = t_N \\ \tau_{t_n} = t_n \cdot 1_{\{G(S(t_n)) \geq E^Q[G(\tau_{t_{n+1}})|S(t_n)])\}} + \tau_{t_{n+1}} \cdot 1_{\{G(S(t_n)) < E^Q[G(\tau_{t_{n+1}})|S(t_n)])\}} \quad for\ n = 0, \ldots, N-1 \end{cases} \tag{2.15}$$

The first approach in chapter 3, the binomial model, uses the idea of value iteration. The underlying stochastic process is a discrete binomial recombining tree, which is ready to work backward in time to calculate both European and American option prices. The American put option can be solved recursively in the binomial model.

$$\begin{cases} P(t_i) = max\{(K - S(t_i))^+, \exp(-r \cdot \Delta t) E^Q[P(t_{i+1})|S(t_i)]\} \quad for\ i = 0, \ldots, N-1 \\ P(t_N) = (K - S(t_N))^+ \end{cases} \tag{2.16}$$

The second approach, Least squares Monte Carlo method (LSM), is to combine policy iteration and Monte Carlo simulation. We are actually looking at a Bermudan option instead of the American option, because we have discrete exercise dates. A Bermudan option initialized at time t_0 has $\mathcal{T}(t_0, t_1, \ldots, t_N)$ exercise dates[14] and for sufficient many exercise dates the Bermudan option approximates the American option well. The method uses regression to calculate the expected continuation value of the simulated path. LSM overcomes the challenge with a pure simulation method by using regression, because it provides an effective way to evaluate a series of conditional expectations. Both methods will be explained in details in chapter 3.

[14] Will also be called decision points

Chapter 3

Classical Numerical Pricing Methods and Closed Form Solutions

In last section we saw that the American put was an example of an option that requires numerical procedures to be priced fairly. The American put is far from the only example of a derivative without a closed form solution. We will look at two classical valuation algorithms for pricing American options in computational finance; the Cox-Ross-Rubinstein (CRR) binomial model and the least squares Monte Carlo method (LSM). The binomial model is an example of a strategy to approximate the option by discretization of the underlying risky asset(s) and the LSM is a method to simulate the underlying risky asset(s). Another popular choice is to solve the free-boundary problem with finite difference methods, but we choose to focus on the two other numerical procedures. The chapter will also investigate valuing exotic multivariate contingent claims. We extend the binomial pricing model (Ekvall, 1996; Boyle, Evnine, and Gibbs, 1989) and LSM to multivariate contingent claims and provide some closed form solutions for exotic European options (Johnson, 1987; Ouwehand, 2006). Therefore the chapter has two purposes; to gain insight into valuation for exotic options and provide benchmarks for the neural networks in the coming chapters.

3.1 Cox Ross Rubenstein Model

The classical binomial model pricing formula or the CRR model presented in this section is inspired by (Cox and Stephen Ross, 1979; Hull, 2017; Björk, 2009). The model will be used for pricing an American put option with one underlying stock and to build the foundation for pricing bivariate contingent claims with the binomial model (Boyle, Evnine, and Gibbs, 1989). The CRR model provides an intuitive and easily implementable model for valuing European and American options. The binomial model has its limitations, because for computational reasons it is not suited for valuing path dependent options or options with several underlying factors. The key difference of the binomial model and the simulation approach is that the binomial model is a discrete time model, where the underlying stochastic process have two possible movements per time-step.

Assume a frictionless market and the underlying stochastic process follow a multiplicative binomial process in discrete time. We work with the financial market $(\Omega, \mathcal{F}, \mathbb{F}, P, B, S_1)$, where the filtration is generated by $\mathbb{F} = \sigma(\mathcal{F}_{t_n})_{n=0,1,\dots,N}$ and the sigma algebra is chosen to be $\mathcal{F} = \mathcal{F}_{t_N}$. It is well known from discrete time arbitrage

theory, that the binomial market model with two assets, where $u > 1 + r > d > 0$ is a arbitrage-free and complete model. The u,d and r describe the evolution of the discrete stochastic process for the stock and the risk-free interest rate on the bank account.

$$B(t_n) = B_0(t_{n-1}) \cdot \exp(\Delta t \cdot r) \quad \textit{where } B(0) = 1 \textit{ and } n = 1, \dots, N$$
$$S_1(t_n) = S_1(0) \prod_{j=1}^{n} Y_j \quad \textit{where } Y_1, Y_2, \dots, Y_N \textit{ are i.i.d., } S_1(0) > 0$$

Note that the interest rate is continuously compounded for computational convenience and the equidistant time step is $\Delta t = T/N$ (section 2.4.3 for notation). We assume that:

$$Y_j = \begin{cases} u & \textit{with probability } p \\ d & \textit{with probability } (1-p) \end{cases}$$

Below, an illustration of a discrete multiplicative binomial process with two time-steps, where the binomial tree recombines[1].

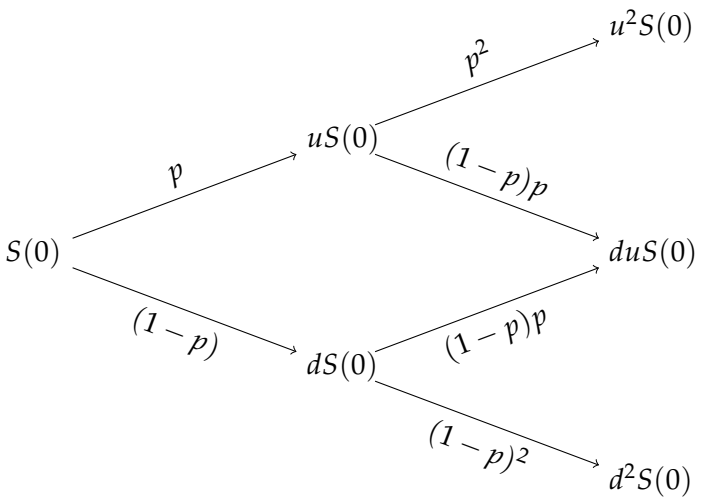


FIGURE 3.1: Price dynamics of binomial lattice model with one underlying risky asset with N=2, $S(0)$ is spot value, p objective probability measure, u and d are realizations of the stochastic variable Y_j

By constructing the discrete process for the stock it is easy to find the equivalent martingale measure.

Definition 3.1. Assume there exists a risk-free asset. A probability measure Q is called a martingale measure if the following condition holds

$$s = \exp(-r\Delta t) \cdot E^Q[S(t + \Delta t)|S(t) = s]$$

Where Δt is a single time-step. (p. 18 (Björk, 2019))

By using definition 3.1 we find the risk neutral measure to be:

$$q = \frac{\exp(r\Delta t) - d}{u - d}$$

[1]Henceforth a binomial recombining tree is referred to as a binomial lattice

The martingale measure q is unique in the binomial lattice model, because the model is complete. Remember the market $(B(t_n), S_1(t_n))$ for the binomial model is arbitrage-free and complete. Hence the pricing formula is given by the risk neutral valuation formula.

Theorem 3.2. ***Risk-neutral valuation formula (RVNF) in discrete time for a T-claim:*** *The arbitrage-free price at t=0 of a T-claim X is given by*

$$\begin{aligned}\Pi(0;X) &= \exp(-r\Delta t \cdot N) \cdot E^Q[X] \\ &= \exp(-r\Delta t \cdot N) \cdot \sum_{n=0}^{N} \binom{N}{n} q^n (1-q)^{N-n} \Phi(s u^n d^{N-n})\end{aligned}$$

Here, Q denotes the martingale measure, $\Pi(0;X)$ *is the price of X to time 0 and* Δt *is a single time step.*

(p. 25 (Björk, 2019))

Theorem 3.2 gives a simple mathematical framework for pricing European options, but the model can readily be extended to American options. American put options for the binomial model will be solved with the dynamic programming approach.

$$\begin{cases} P(t_i) = max\{(K - S(t_i))^+, \exp(-r \cdot \Delta t) E^Q[P(t_{i+1})|S(t_i)]\} & for\ i = 0, \dots, N-1 \\ P(t_N) = (K - S(t_N))^+ \end{cases}$$

Note that the binomial lattice model has a recursive structure, where the one-step transition probabilities are the same in each node. The idea is to, at each decision point, either exercise to gain the intrinsic value if it is greater than the expected continuation value or hold the option alive for another period. The dynamic choice to exercise or to keep the option alive gives a exercise barrier b (Figure 3.2).

To value an American put option we lay out all the possible paths of the stock in the tree based on $S_1(0), \sigma$ and T. To construct the tree we need to specify the number of equidistant time-steps Δt ($\Delta t = \frac{T}{N}$ *where* $N = No.\ of\ steps$) for the tree, where for each step we add another possible value for the stock. We only add one more possibility for each time-step because the tree recombines[2]. Figure 3.2 is an example of a constructed tree, where the value of the option is also included by color. The decision to exercise is marked with a triangle and continuation is marked with a circle.

The u and d are chosen to be the reciprocal of each other and they are essentially functions of the volatility. The d and u are chosen so they match the first and second moment of the lognormal distribution, because Black-Scholes theory assume the future price of the stock is lognormal distirbuted (Detailed explanation in Appendix C.1).

$$u = \exp(\sigma\sqrt{\Delta t}) \quad d = \exp(-\sigma\sqrt{\Delta t})$$

For valuing an American put option, we value the exercise value at maturity (time T) for all possible outcomes for the price process at maturity. Then we use backward induction/dynamic programming where we compare the intrinsic value with the expected continuation value (equation (3.1)), where we choose the maximum of these two. The comparison will be applied for every node in each decision point $(t_n)_{n=0,1,\dots,N-1}$ and all the way back in time to the initialization date. Through this

[2] $(1+n)$ possible stock values after n steps

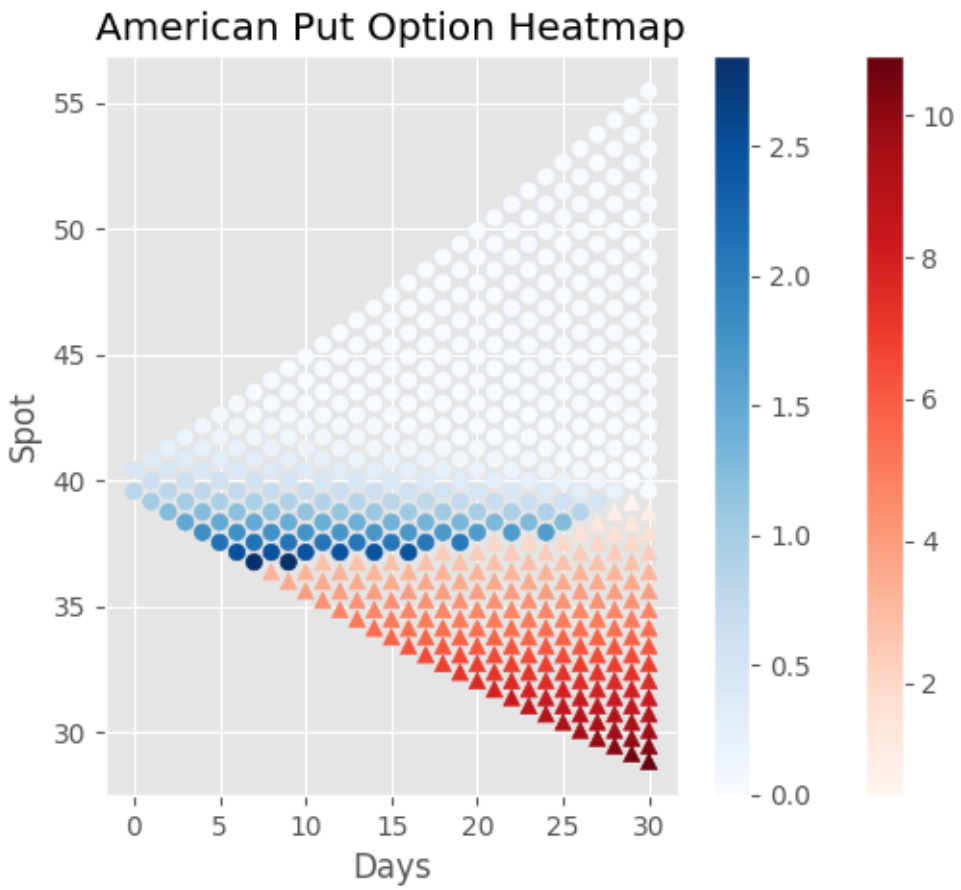


FIGURE 3.2: A valuation tree of an American put option price based on the binomial model, where the color indicates the value and the dots mark the continuation nodes. The parameters are S(0)=40, N=30, $\Delta t = 1$ day, K=40 and u=1.0106

procedure we derive present value of the American option in the discrete time model. One design decision is to choose the number of time-steps considering a trade-off between computational speed and accuracy. Figure 3.3 illustrates that the option value stabilizes making time-steps smaller for an option with 1 year to maturity. The precision for the algorithm also increases by increasing the number of time-steps.

We have seen central concepts arbitrage and completeness from continuous time in action for the discrete time setup. The paper (Cox and Stephen Ross, 1979) which introduced the binomial model to option pricing came after the Black-Scholes model described in section 2 (Black and Scholes, 1973). The main reason for developing a binomial model is that the discrete time approach gives a simplified model in terms of the mathematics and highlights the essential concepts in arbitrage theory. You can argue that the simpler mathematics in this model makes the binomial pricing method more instructive and clear. Besides being easier to understand for non-mathematician, it works nicely with other options than the European options like American options.

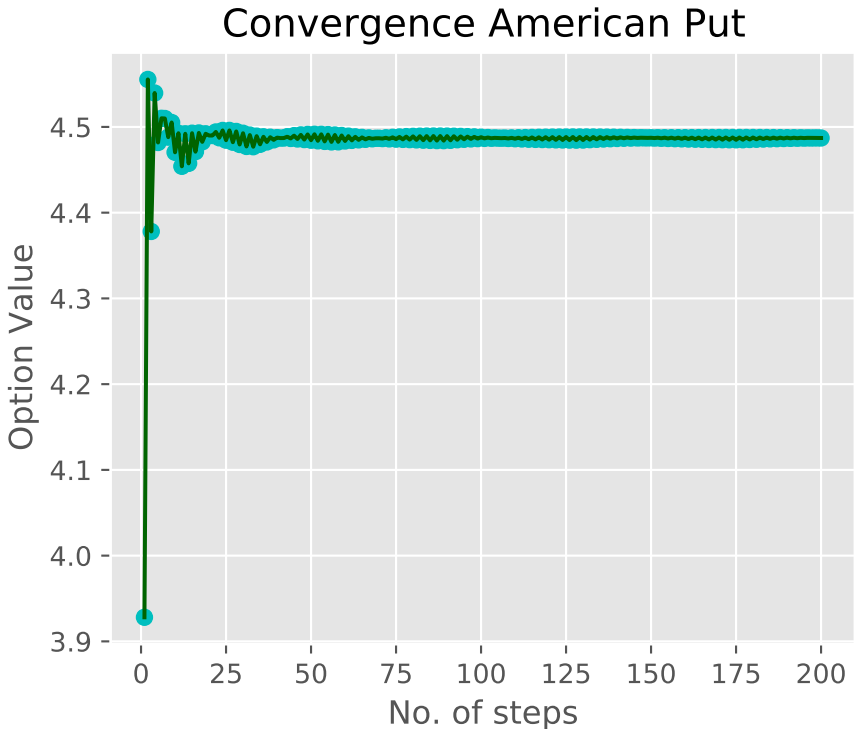


FIGURE 3.3: Price for a American put option based on the binomial model, where the independent variable is the number of time-steps. Parameters are $T = 1$, $K = 40$, $S(0)$=40, σ=0.2, and $r = 0.06$.

Even though we assume the stock price moves at discrete jumps instead of log-normal returns in the continuous time model, it can actually be shown that the CRR binomial model will converge to the Black-Scholes model. Hence the binomial pricing model will be equivalent with the continuous time analytical pricing model derived by Fischer Black and Myron Scholes in the limit for European options.

The path-independence payoff for the American put makes the tree recombining, so there is only N+1 terminal nodes at maturity. If the derivative was path-dependent e.g. an Asian option, then we have a non-recombining tree and 2^N terminal nodes are needed. This is computational inefficient, which explains that the binomial model should not be used for path-depending derivatives. We will show below, that the intuitive CRR model can be extended to higher dimensions, but the model suffers from the curse of dimensionality.

3.2 Binomial Lattice Model for Multivariate Contingent Claims

We follow the approach in (Boyle, Evnine, and Gibbs, 1989), henceforth called the BEG method. The BEG method is a natural extension of the Cox Ross Rubinstein model (section 3.1) for multivariate contingent claims. The idea, as in the one dimensionel case, is to approximate the system of underlying processes[3] with a discrete multivariate binomial lattice. The advantage is that for exotic options like the basket options the valuation of European put options is readily extended to American put options.

The BEG method has its limitation in terms of number of underlying assets and for path dependent options (see section 3.1), but it is very intuitive. The problem with increasing the number of underlying assets is that the number of one-step transition at each node is 2^d and the total number of terminal nodes after N steps is $(N+1)^d$ for path-independent derivatives. It means that the computational resources become an issue for high dimensional problems with the discrete time model approach. This makes the BEG method undesirable for options with many underlying assets. Another problem with three or more underlying assets is that some one-step transition probabilities can turn out negative with the BEG method, which makes the model nonsensical in those cases. For those reasons we choose to focus on the bivariate contingent claim i.e. $d = 2$.

The model we want to approximate is the bivariate lognormal distribution, because we assume the Black-Scholes model to describe the evolution of the two risky assets (section 2.2). We restrict ourselves to the Black-Scholes assumptions given in section 2.3. Hence, under risk neutral measure the SDE for the risky assets are:

$$dS_i(t) = S_i(t)r(t)dt + S_i(t)\sigma_i(t)dW^Q(t) \quad for\ i = 1,2$$

We divide the time from inception to maturity into N equidistant intervals with length Δt. We want the jump distribution to approximate the continuous time bivariate lognormal distribution. Each time interval has a jump size defined in terms of the volatility and the length of the interval:

$$u_i = \exp(\sigma_i\sqrt{\Delta t}) \quad and \quad u_i \cdot d_i = 1 \quad for\ i = 1,2$$

The u_i and d_i are the multiplication factors for the i'th stock, where the former is a jump up and the latter is a jump down factor for the stock. What is the probability that the stock jumps up or down? The probabilities are chosen such that the characteristics functions are equal for small time steps Δt (p. 245-246 in (Boyle, Evnine, and Gibbs, 1989) for details). The probabilities for the model with two underlying

[3] In Black-Scholes theory the stochastic process evolution for the stock is described by the GBM

risky assets are:

$$\begin{aligned} q_1 = q_{uu} &= \frac{1}{4}\left(1+\rho+\sqrt{\Delta t}(\frac{\mu_1}{\sigma_1}+\frac{\mu_2}{\sigma_2})\right) \\ q_2 = q_{ud} &= \frac{1}{4}\left(1-\rho+\sqrt{\Delta t}(\frac{\mu_1}{\sigma_1}-\frac{\mu_2}{\sigma_2})\right) \\ q_3 = q_{du} &= \frac{1}{4}\left(1-\rho+\sqrt{\Delta t}(-\frac{\mu_1}{\sigma_1}+\frac{\mu_2}{\sigma_2})\right) \\ q_4 = q_{dd} &= \frac{1}{4}\left(1+\rho+\sqrt{\Delta t}(-\frac{\mu_1}{\sigma_1}-\frac{\mu_2}{\sigma_2})\right) \end{aligned} \tag{3.1}$$

The correlation ρ between the two assets are assumed to be constant and $\mu_i = r - \frac{1}{2}\sigma_i^2$. We have illustrated below the evolution of binomial lattice model with two underlying risky assets, where we see that the number of nodes at maturity is $(1+N)^d$.

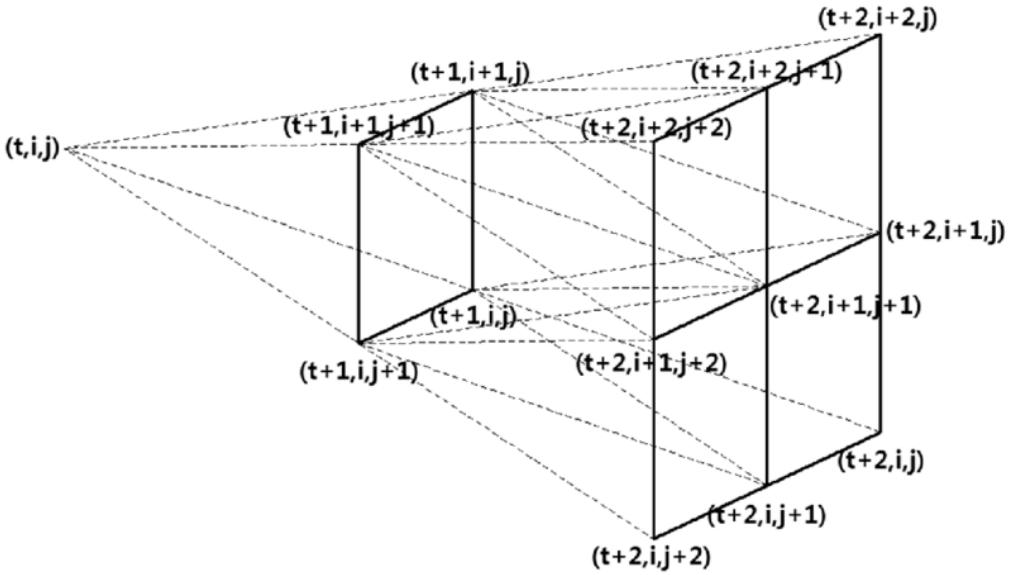


FIGURE 3.4: Evolution of binomial lattice model with two underlying risky assets, where t is time, i is number of up movements for S_1, and j is number of up movements for S_2. Figure is from the paper (Kim, Lee, and An, 2020)

After the construction of the evolution of the underlying assets, we can, like in the one dimensional CRR model, recursively working backwards in the multidimensional binomial lattice. For the European option the recursive formula is:

$$J_{i_1,\ldots,i_d}(t_n) = \exp(-r\Delta t)\left(q_1 J_{i_1+1,\ldots,i_d+1}(t_{n+1}) + \cdots + q_{2^d} J_{i_1,\ldots,i_d}(t_{n+1})\right)$$

Where J is the value of the option to time t_n and i is the number of times the underlying spot price at inception has been multiplied by u. For the American option the recursive formula is:

$$J_{i_1,\ldots,i_d}(t_n) = \max\{\Phi(t_n, \boldsymbol{S}(t_n)), \exp(-r\Delta t)\left(q_1 J_{i_1+1,\ldots,i_d+1}(t_{n+1}) + \cdots + q_{2^d} J_{i_1,\ldots,i_d}(t_{n+1})\right)\}$$

With the recursive formulas we can valuate multivariate contingent claims for a variety of exotic options including the American put option with several underlying risky assets.

The computational required resources increases with the number of underlying assets for the binomial lattice methods. For the BEG method we have 2^d number of one-step transition probabilities. To reduce the number of transition probabilities, it had been tried in (Ekvall, 1996), to set all the probabilities equal to 0.5 and then calculate the jump sizes. This is the reversed process compared to the BEG method, where the jump sizes are set first and then one-step probabilities are calculated. Choosing one-step probabilities and then the jump sizes has overcome the issue with negative probabilities for higher dimensions and in (Ekvall, 1996) the algorithm also seems to have faster convergence. The reserved algorithm of BEG is called the NEK method.

The NEK and BEG approaches are both good in terms of both accuracy and computational speed for low dimensional option problems, but both still suffer from the curse of dimensionality. The simulations methods on the other hand, have somewhat of an advantage for higher dimensions. We choose to present the BEG approach, because it is a natural extension of the already presented CRR model, which we already build intuition upon. The key for the BEG approach is that we approximate a multivariate lognormal distribution with a discrete jump distribution.

3.3 Least Squares Monte Carlo Method

We have seen that the binomial model is not suitable for path-dependent contingent claims like an Asian option and high dimensional multivariate contingent claims. Simulation methods overcome somewhat this issue. One of the first Monte Carlo methods for option pricing were to use pure simulation techniques. These methods overcame the issue of path-dependent option, but they still suffered the curse of dimensionality. To solve the dimensionality problem the LSM was born, where the idea is to combine simulation with regression.

A pure simulation technique has three ingredients:

- A simulation based on the assumption of the underlying asset(s) distribution to price potential future prices
- From the simulation of the underlying asset(s) use the contract function to calculate the cash flow
- Discount back the cash flow to present value and average over the simulated paths

The approach is suitable for the Asian option, because by simulation you have the whole path and averaging is straightforward. On the other hand, the binomial model is computationally fast and accurate for American options, because by discretization of the underlying stock the algorithm gives an effective way to get intrinsic and expected continuation values. The pure simulation method would not be ideal for American options, because at each decision point for the American option, the pure simulation would need to simulate a new set of paths to estimate the expected continuation value. For reference of the pure simulation method see chapter 10 in (OVERHAUS et al., 2007).

Remember, we approximate the American put option with a Bermudan put option. The simulatation methods use the setup in section 2.4.3. Again, we use dynamic programming to calculate the price of the American put option. The problem with the pure simulation approach is the computational burden to evaluate the expected continuation value at each exercise date. The Longstaff and Schwartz LSM algorithm overcomes the exponential growing computational burden in pure Monte Carlo simulation by using regression to calculate the expected continuation value.

3.3.1 The Algorithm

To solve the optimal stopping problem numerically, we set up the mathematical framework for solving the problem inspired by (Clément, Lamberton, and Protter, 2001). The optimal problem in discrete time is:

$$\sup_{\tau \in \mathcal{T}(0,\dots,T)} E^{Q}[G(S(\tau))] \tag{3.2}$$

where $\mathcal{T}(0,\dots,T)$ is a class of all $(0,\dots,T)$-valued stopping times and $\Big(S(0), S(t_1), \dots, S(t_N) = S(T)\Big)$ is the underlying stochastic Markovian process describing e.g. stock price,

volatility, average stock price, etc. The Markov chain $(S_{t_n})_{n=0,\dots,N}$ has state space $(E, \mathcal{E})$. Remember the snell envelope is given by:

$$U(t_n) = \operatorname*{ess\,sup}_{\tau \in \mathcal{T}(t_n,\dots,t_N)} E^Q[G(S(\tau))|\mathcal{F}_n]$$

The Markov property implies that:

$$E^Q[G(S(\tau_{t_{n+1}}))|\mathcal{F}_{t_n}] = E^Q[G(S(\tau_{t_{n+1}}))|S(t_n)] = f(S(t_n))$$

and we assume the initial state to be $S(0) = s$ and deterministic. We solve equation (3.2) by the theory presented in section 2.4.3. Hence, the dynamic programming principle on the optimal policy is:

$$\begin{cases} \tau_{t_N} = t_N \\ \tau_{t_n} = t_n \cdot 1_{\{G(S(t_n)) \geq E^Q[G(S(\tau_{t_{n+1}}))|S(t_n)])\}} + \tau_{t_{n+1}} \cdot 1_{\{G(S(t_n)) < E^Q[G(S(\tau_{t_{n+1}}))|S(t_n)])\}} \end{cases} \quad for\ n = 0, \dots, N-1 \tag{3.3}$$

Equation (3.3) involves many conditional expectations. Hence, we need an effective algorithm for evaluating them. The pure simulation methods were expensive, because the paths required for evaluating the series of conditional expectations increases exponentially with decision points (chapter 10 (OVERHAUS et al., 2007)). The solution suggested e.g. in (Longstaff and Schwartz, 2001; Tsitsiklis and Roy, 1999) is to use the cross-sectional information in the simulated paths. The information is used for least squares regression.

The LSM algorithm approximates the expected continuation value with respect to $S(t_n)$ by orthogonal projection on the state-space generated by finite number of basis functions of $S(t_n)$. Define the sequence $(e_j(s))_{j\geq 1}$ of real measureable functions defined on the state space $(E, \mathcal{E})$. Assume:

- The sequence $(e_j(S(t_n)))_{j\geq 1}$ is total in $L^2(\sigma(S(t_n)))$ for $n = 1, \dots, N-1$.
- if $\sum_{j=1}^m \lambda_j e_j(S(t_n)) = 0$ *a.s.* then $\lambda_j = 0$ for $n = 1, \dots, N-1$, $m \geq 1$ and $j = 1, \dots, m$

(p. 3 (Clément, Lamberton, and Protter, 2001))

By defining the vector space generated by $(e_j(s))_{j=1,\dots,m}$ we denote the orthogonal projection from $L^2(\Omega)$ onto the vector space by $P^m_{t_n}$. We approximate the expected continuation value by the projection:

$$\begin{cases} \tau^{[m]}_{t_N} = t_N \\ \tau^{[m]}_{t_n} = t_n \cdot 1_{\{G(S(t_n)) \geq P^m_{t_n}(G(S(\tau^{[m]}_{t_{n+1}})))\}} + \tau^{[m]}_{t_{n+1}} \cdot 1_{\{G(S(t_n)) < P^m_{t_n}(G(S(\tau^{[m]}_{t_{n+1}})))\}} \end{cases} \quad for\ n = 0, \dots, N-1$$

Where:

$$P^m_{t_n}(G(S(\tau^{[m]}_{t_{n+1}}))) = \alpha^m(t_n) \cdot e^m(S(t_n)) \quad for\ n = 1, \dots, N-1$$

and the $\cdot$ on the right hand side is the inner product in $\mathbb{R}^m$ and $e^m = (e_1, \dots, e_m)$. By solving the above dynamic programming equation the time 0 approximate price of the Bermudan option is:

$$U^m(0) = \max\{G(S(0)), E^Q[G(S(\tau^{[m]}_{t_1}))]\}$$

Where $G(S(0))$ is deterministic by assumption, but $E^Q[G(S(\tau_{t_1}^{[m]}))]$ needs to be evaluated numerically. In the LSM algorithm the methods for evaluation the projection is Monte Carlo simulation.

From the assumption about the distribution of the underlying state space we simulate K independent paths $S^{(1)}(t_n), S^{(2)}(t_n), \ldots, S^{(k)}(t_n), \ldots, S^{(K)}(t_n)$ of the Markov chain $S(t_n)$ with gain process $G(S^{(k)}(t_n))$ for $k = 1, \ldots, K$ and $n = 0, \ldots, N$. The least squares estimator $\alpha^{(m,K)}(t_n) \in \mathbb{R}^m$ for the simulated paths is:

$$\alpha^{(m,K)}(t_n) = \underset{a \in \mathbb{R}^m}{\operatorname{argmin}} \sum_{k=1}^{K} \left(G(S^{(k)}(\tau_{t_{n+1}}^{k,m,K})) - a \cdot e^m(S^{(k)}(t_n)) \right)^2 \quad for\ n = 1, \ldots, N-1 \tag{3.4}$$

Where the recursively estimated stopping time is defined by:

$$\begin{cases} \hat{\tau}_{t_N}^{k,m,K} = t_N \\ \hat{\tau}_{t_n}^{k,m,K} = t_n \cdot 1_{\{G(S^{(k)}(t_n)) \geq \alpha^{m,K}(t_n) \cdot e^m(S^{(k)}(t_n))\}} + \hat{\tau}_{t_{n+1}}^{k,m,K} \cdot 1_{\{G(S^{(k)}(t_n)) < \alpha^{m,K}(t_n) \cdot e^m(S^{(k)}(t_n))\}} \end{cases} \quad for\ n = 0, \ldots, N-1 \tag{3.5}$$

An example of polynomial regression on each decision point is illustrated in figure 3.5, where the blue line is the estimated expected continuation value. From following the optimal stopping strategy by dynamic programming we derive the approximation for $U^m(0)$ from the simulated paths. The time 0 approximate price of the Bermudan option is:

$$U^{m,K}(0) = \max\{G(S(0)), \frac{1}{K} \sum_{k=1}^{K} G(S^{(k)}(\hat{\tau}_{t_1}^{k,m,K}))]\}$$

3.3.2 American Put

The optimal stopping problem for an American put

$$P(0,s) = \sup_{\tau \in \mathcal{T}(0,\ldots,T)} E_s^Q[\exp(-r\tau) \cdot \max\{K - S(\tau), 0\}] \tag{3.6}$$

can be solved with the LSM algortihm. The stock values are modeled via Black-Scholes theory[4], hence, the simulated evolution for the stock under the risk neutral valuation is given by:

$$S_i(t) = S_i(0) \cdot \exp\left(\sum_{j=1}^{d} (\sigma_{i,j} W_j(t) - \frac{1}{2}\sigma_{i,j}^2 t) + rt \right) \quad for\ i = 1, \ldots, d$$

The stock paths are simulated from inception up to maturity with N+1 exercise dates. The focus in this section is on a univariate contingent claim and for convenience we assume the risk-free interest rate and the volatility are constants. As in the binomial model, we first construct the paths and then work backwards from maturity to inception at each exercise date to decide the optimal stopping time.

From previously, we know the dynamic programming principle on optimal policy gives the first optimal stopping time and at each decision point we regress the

[4]Illustration of generated path for the GBM in figure 2.2

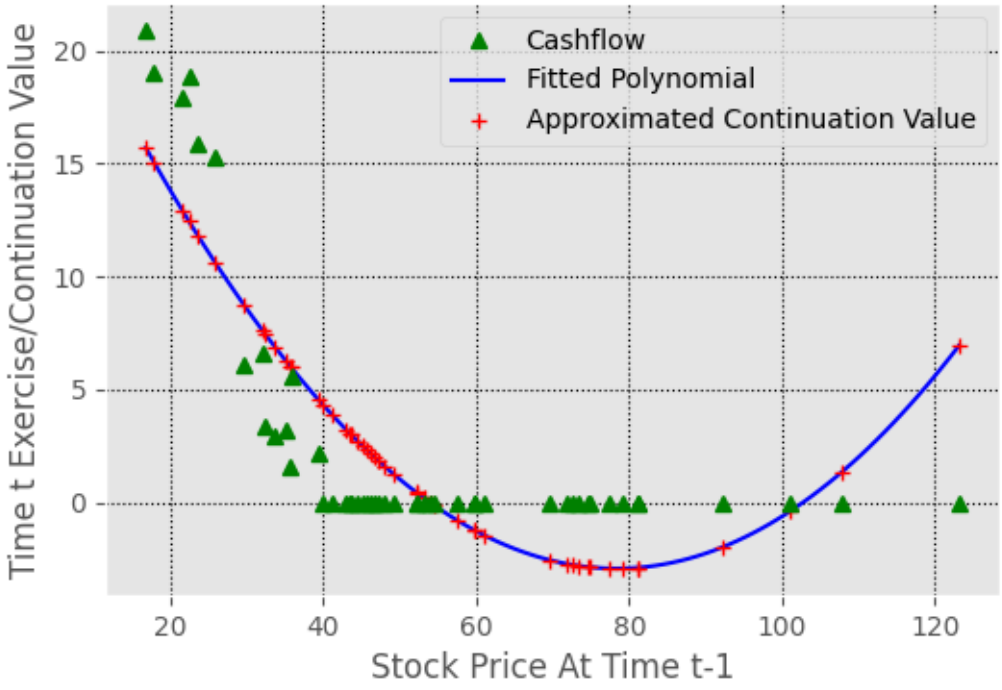


FIGURE 3.5: By zooming in on a specific point of time in the dynamic programming approach, we see how the algorithm regress the continuation value. In the illustration we used second order polynomial regression

expected payoff by continuation of the contract and compare it to the intrinsic value. The dependent variable in the regression is the expected value of continuation and the independent variables are a set of orthogonal basis functions in $L^2(\sigma(S(t_n)))$ of the simulated paths. Typical choices for basis functions could be weighted Laguerre -, Hermit -, and Jacob polynomials. The weighted Laguerre polynomial is given by:

$$\begin{aligned} e_0(S) &= \exp(-S/2) \\ e_1(S) &= \exp(-S/2)(1-S) \\ e_2(S) &= \exp(-S/2)(1-2S+S^2/2) \\ &\vdots \\ e_j(S) &= \exp(-S/2)\frac{\exp(S)}{j!}\frac{d^j}{dS^j}(S^j\exp(-S)) \end{aligned}$$

This kind of regression is a nonlinear expansion of the linear model[5]. We define regressed conditional expectation by:

$$\Psi(S(t_n);\alpha) = \sum_{j=0}^{p} \alpha_j \cdot e_j(S(t_n))$$

[5]Prediction is a linear function of the coefficients

where α is the coefficient for the regression, e is the basis function, where the argument is the underlying Markovian process S. By using this iterative method, we arrive at the pathwise optimal stopping policy, where in figure 3.6 an example of pathwise optimal stopping times are shown. The figure illustrates that the option only can be exercised once, hence, the gray lines after the triangles.

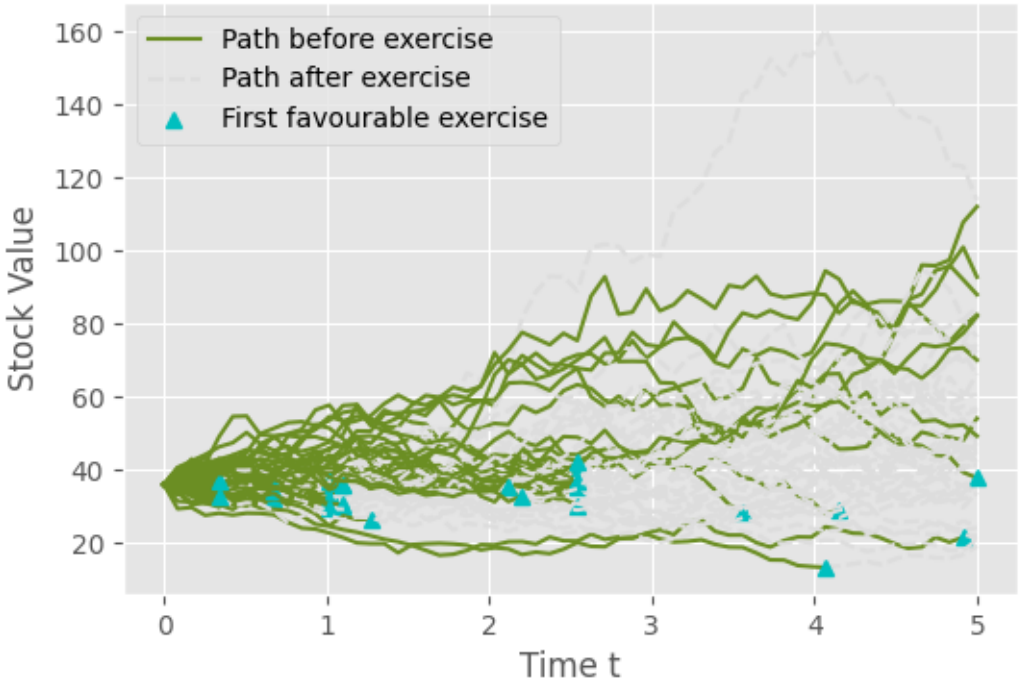


FIGURE 3.6: The optimal stopping decisions by the Least Squares Monte Carlo Method

Note that for practical implementation the LSM is only considered in the money path. Hence, equation (3.4) is changed to:

$$\alpha^{(m,K)}(t_n) = \underset{a\in\mathbb{R}^m}{\operatorname{argmin}} \sum_{k=1}^{K} 1_{\{G(S^{(k)}(t_n))>0\}} \left(G(S^{(k)}(\tau_{t_{n+1}}^{k,m,K})) - a \cdot e^m(S^{(k)}(t_n)) \right)^2 \quad for\ n = 1, \ldots, N-1$$

Therefore, only the in-the-money (ITM) paths are used for regression. Using only the ITM paths improves the algorithm, because the option value can never be less than zero, but including the other paths can yield negative results. The two articles (Longstaff and Schwartz, 2001; Tsitsiklis and Roy, 1999) have some differences in their approaches. The LSM considers only ITM and uses policy iteration instead of value iteration. Furthermore, in (Tsitsiklis and Roy, 1999) they set $U(t_n) = \Psi(S(t_n); \alpha)$ instead of $U(t_n) = U(t_{n+1})$ as in the LSM. These design choices make the LSM the most used method and it has the highest accuracy.

Note, for our computational implementation we use the same paths for decision making and valuation, this gives a biased high, but the difference in practice is not noticeable. In general. it leads the LSM to a biased low, because of the sub-optimal

exercise strategy[6]. Therefore, an upper bound can be computed for the option price, which is done in (Andersen and Broadie, 2004). We wrap up the LSM section by looking at how the method can be used for basket options.

3.3.3 LSM Extension to Multivariate Contingent Claims

For pricing of bivariate and multivariate contingent claims, we have to account for correlation between assets. The correlation matrix $\boldsymbol{\rho}$ is given by:

$$\boldsymbol{\rho} = \begin{pmatrix} \rho_{1,1} & \rho_{1,2} & \cdots & \rho_{1,d} \\ \rho_{2,1} & \rho_{2,2} & \cdots & \rho_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{d,1} & \rho_{d,2} & \cdots & \rho_{d,d} \end{pmatrix} = \begin{pmatrix} 1 & \rho_{1,2} & \cdots & \rho_{1,d} \\ \rho_{2,1} & 1 & \cdots & \rho_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{d,1} & \rho_{d,2} & \cdots & 1 \end{pmatrix}$$

The correlation between assets is given by $(\rho_{i,j})_{i \neq j}$ and each asset has correlation 1 with itself. We assume that $(\rho_{i,j})_{i \neq j} \in (-\frac{1}{d-1}, 1]$, because then the correlation matrix is a real, symmetric, and positive definite. Hence, Cholesky factorization can be utilized $\boldsymbol{\rho} = \mathbf{L}\mathbf{L}^T$ where $\mathbf{L}$ is a lower triangular matrix. From the decomposition it is easy to simulate correlated assets in d-dimensional Black-Scholes model:

$$dS_i(t) = S_i(t)rdt + S_i(t)\sigma_i \mathbf{L}_{i,:} d\boldsymbol{W}(t) \quad i \in \{1, 2, \ldots, d\}$$

Where $\boldsymbol{W}$ lies in $\mathbb{R}^d$ and $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_d)$ is vector of volatilities.

By simulating the paths according to the Black-Scholes dynamic, it is straightforward to apply the LSM, because it follows the same algorithm just with a different contract function.

Note in the next section, it will be more convenient to work with the covariance matrix Σ and the covariance matrix is given by:

$$\Sigma = D(\boldsymbol{\sigma})\boldsymbol{\rho}D(\boldsymbol{\sigma})$$

The Cholesky factorization is then:

$$\Sigma = \mathbf{A}\mathbf{A}^T$$

[6]The LSM algorithm gives a lower bound given in appendix C.3

3.4 Closed Form Solutions for European Exotic Options

Most exotic options require numerical methods, but in some special cases there exist a closed form solution. We will look at some of them in this section, where the purpose is to provide benchmarks for the numerical methods. Furthermore, we explore the boundaries of closed form solutions and show applications of martingale theory. Throughout the financial market and model assumptions given in section 2.2 will be assumed. We derive closed form solutions to European call and put options depending on several variables, for simplicity we will focus on pricing options with two or three underlying stocks. The section is inspired by the article (Ouwehand, 2006). The exotic contingent claims we will consider are the geometric mean -, maximum - and minimum call option.

3.4.1 Geometric Mean Basket Call Option

For a geometric mean basket call option the contract function is given by:

$$\Phi(\boldsymbol{S}(T)) = \max\{(\prod_{i=1}^{d} S_i(T))^{\frac{1}{d}} - K, 0\}$$

The key to derive a closed form solution is the known result that the sum of normal random variables is multivariately normally distributed. This implies that the product of lognormal random variables are multivariately lognormally distributed, since:

$$\exp(x+y) = \exp(x)\cdot\exp(y)$$
$$\text{and}$$
$$X \sim \mathcal{N}(\mu,\sigma^2) \Rightarrow Y = \exp(X) \sim LN(\mu,\sigma^2)$$

Remember the assumption in section 2.2 that the stock price process follows a GBM, hence:

$$(\prod_{i=1}^{d} S_i(T))^{\frac{1}{d}} = (\prod_{i=1}^{d} S_i(0))^{\frac{1}{d}} \exp\left((r - \frac{1}{2d}\sum_{i=1}^{d}\sigma_i^2)T + \frac{1}{d}\sum_{i=1}^{d}\sigma_i W_i(T)\right) \tag{3.7}$$

Define:

$$\tilde{\sigma} = \frac{1}{d}\sqrt{\sum_{i=1}^{d}\sigma_i^2 + 2\sum_{i\neq j}\rho_{i,j}\sigma_i\sigma_j}$$
$$F = (\prod_{i=1}^{d} S_i(0))^{\frac{1}{d}} \exp\left((r - \frac{1}{2d}\sum_{i=1}^{d}\sigma_i^2)T + \frac{1}{2}\tilde{\sigma}^2\cdot T\right)$$
$$\epsilon = \frac{\frac{1}{d}\sum_{i=1}^{d}\sigma_i W_i(T)}{\tilde{\sigma}\sqrt{T}} \sim \mathcal{N}(0,1)$$

and rewrite equation (3.7) by above definitions:

$$(\prod_{i=1}^{d} S_i(T))^{\frac{1}{d}} = F\cdot\exp\left(-\frac{1}{2}\tilde{\sigma}^2\cdot T + \tilde{\sigma}\sqrt{T}\epsilon\right)$$

This expression is one dimensional and is the standard GBM solution with zero drift and spot F. This has a known solution with Black-Scholes theory (section 2.3) and the geometric mean call option has price

$$\Pi(t,\mathcal{X}) = \exp(-r\cdot(T-t))\Big(FN(d_1) - KN(d_2)\Big)$$

where $d_1 = \frac{\ln(\frac{F}{K})+\frac{1}{2}\tilde{\sigma}^2 T}{\tilde{\sigma}\cdot\sqrt{T}}$ and $d_2 = d_1 - \tilde{\sigma}\sqrt{T}$

The fact that the sum of normal random variables is multivariately normally distributed makes the geometric mean option easy to price in the Black-Scholes model, because the high dimensional problem can be treated as a 1-dimensional problem.

3.4.2 Options on the Maximum or the Minimum of Several Assets

Here we restrict ourselves to consider the case with three underlying stocks like in (Ouwehand, 2006), but the formula can be generalized to higher dimensions. The contract functions we consider are:

- Best of assets or cash: $\Phi(S(T)) = \max\{S_1, S_2, \ldots, S_d\}$
- Best of assets or cash: $\Phi(S(T)) = \max\{S_1, S_2, \ldots, S_{d-1}, K\}$
- Call on maximum: $\Phi(S(T)) = \max\{\max(S_1, S_2, \ldots, S_{d-1}) - K, 0\}$
- Call on mininum: $\Phi(S(T)) = \max\{\min(S_1, S_2, \ldots, S_{d-1}) - K, 0\}$

Assume WLOG d=4 to avoid cumbersome calculations and notation. The section will heavily use the martingale framework developed in section 2.2 and stochastic calculus (Appendix A) to value these exotic options. The key is to choose the numéraire to a risky asset instead of the bank account. By results from section 2.2 the processes are still Q-martingales given the numéraire is strictly positive. Under the assumption of an arbitrage-free and complete market it follows from risk neutral valuation formula:

$$S_0(t)E_t^{Q_0}[\frac{X}{S_0(T)}] = S_1(t)E_t^{Q_1}[\frac{X}{S_1(T)}]$$

This shows that changing the numéraire does not change the price of the derivative.

3.4.2.1 Best of Assets or Cash

The best of assets derivative will provide the foundation for pricing best of assets or cash and call on maximum or minimum options. The best of assets derivative pay out the price of the most valuable asset. The idea is then to set up 4 derivatives, where each derivative only gives a payout if the underlying stock is the most valuable out of the four stocks. This can be written with an indicator, where the payoff for the i'th asset is:

$$S_i(T)\cdot 1_{\{S_i(T)>S_j(T):i\neq j\}} \quad i = \{1,2,3,4\} \tag{3.8}$$

From equation (3.8) we see that the i'th asset is only different from zero, when it is the most valuable asset. The fair price for best of assets option $\Pi_{max}(t,\mathcal{X})$ is then given as the sum of the 4 derivatives. Hence, the best of asset derivative can be found by appling RNVF (proposition 2.13) for each of them.

Let us first consider i=1 and we set S_1 to be the numéraire asset with martingale measure $\mathbb{Q}_1$. Then by RNVF:

$$\begin{aligned}\Pi_1(t,\mathcal{X}) &= S_1(t)E_t^{\mathbb{Q}_1}[1_{\{S_1(T)>S_2(T),S_1(T)>S_3(T),S_1(T)>S_4(T)\}}] \\ &= S_1(t)Q_1[\ln(\frac{S_2(T)}{S_1(T)})<0,\ln(\frac{S_3(T)}{S_1(T)})<0,\ln(\frac{S_4(T)}{S_1(T)})<0]\end{aligned} \tag{3.9}$$

From above discussion the derivative price can be found by cycling through the 4 derivatives. Equation (3.9) can be evaluated, therefore we seek to evaluate the probability under the Q-martingale measure. By Ito's lemma (see A.1) the discounted process with stock j is:

$$d(\frac{S_i(t)}{S_j(t)}) = \frac{S_i(T)}{S_j(T)}\cdot(a_i-a_j)dW^{Q_j}(t)$$

where $dW^{Q_j}(t)$ is standard d-dimensional Q_j-Wiener process and a_i is the i'th row of the matrix $\mathbf{A}$ given in section 3.3.3.

From the known solution of the GBM we have the distribution.

$$\ln(\frac{S_i(T)}{S_j(T)}) \sim \mathcal{N}\left(\ln(\frac{S_i(t)}{S_j(t)}) - \frac{1}{2}\sigma_{i/j}^2\cdot(T-t), \sigma_{i/j}^2(T-t)\right)$$

where $\sigma_{i/j} = (a_i - a_j)$ and $\sigma_{i/j}^2 = \sigma_i^2 + \sigma_j^2 - 2\rho_{ij}\sigma_i\sigma_j$.

Besides using the definition for d_1 and d_2 in proposition 2.16 we define:

$$d_1^{i/j} = \frac{1}{\sigma_{i/j}\cdot\sqrt{T-t}}\cdot\left(\ln(\frac{S_i(t)}{S_j(t)}) + \frac{1}{2}\sigma_{i/j}^2\cdot(T-t)\right)$$
$$d_2^{i/j} = d_1^{i/j} - \sigma_{i/j}\sqrt{T-t}$$

The correlation between $\frac{S_i(T)}{S_k(T)}$ and $\frac{S_j(T)}{S_k(T)}$ is:

$$\begin{aligned}\rho_{ij,k} &:= \frac{(a_i-a_k)\cdot(a_j-a_k)}{||a_i-a_k||\cdot||a_j-a_k||} \\ &= \frac{\rho_{ij}\sigma_i\sigma_j - \rho_{ik}\sigma_i\sigma_k - \rho_{kj}\sigma_k\sigma_j + \sigma_k^2}{\sqrt{(\sigma_i^2+\sigma_k^2-2\rho_{ik}\sigma_i\sigma_k)\cdot(\sigma_j^2+\sigma_k^2-2\rho_{jk}\sigma_j\sigma_k)}}\end{aligned}$$

Hence:

$$Q_1[\ln(\frac{S_2(T)}{S_1(T)})<0,\ln(\frac{S_3(T)}{S_1(T)})<0,\ln(\frac{S_4(T)}{S_1(T)})<0] = N_3(-d_2^{2/1},-d_2^{3/1},-d_2^{4/1},\rho_{23,1},\rho_{24,1},\rho_{34,1})$$

Cycling through each derivative, we get:

$$\begin{aligned}\Pi_{max}(t,\mathcal{X}) &= S_1(t)N_3(-d_2^{2/1},-d_2^{3/1},-d_2^{4/1},\rho_{23,1},\rho_{24,1},\rho_{34,1}) \\ &+ S_2(t)N_3(-d_2^{1/2},-d_2^{3/2},-d_2^{4/2},\rho_{13,2},\rho_{14,2},\rho_{34,2}) \\ &+ S_3(t)N_3(-d_2^{1/3},-d_2^{2/3},-d_2^{4/3},\rho_{12,3},\rho_{14,3},\rho_{24,3}) \\ &+ S_4(t)N_3(-d_2^{1/4},-d_2^{2/4},-d_2^{3/4},\rho_{12,4},\rho_{13,4},\rho_{23,4})\end{aligned} \tag{3.10}$$

We can extend the above result to best of assets and cash by letting $S_4(t) = K\exp(-r(T-$

$t)$), where K does not have any volatility and is also independent of the other assets, hence, equation (3.10) becomes:

$$\begin{aligned}\Pi_{max}(t,\mathcal{X}) &= S_1(t)N_3(-d_2^{2/1},-d_2^{3/1},d_1^1,\rho_{23,1},\rho_{24,1},\rho_{34,1})\\ &+ S_2(t)N_3(-d_2^{1/2},-d_2^{3/2},d_1^2,\rho_{13,2},\rho_{14,2},\rho_{34,2})\\ &+ S_3(t)N_3(-d_2^{1/3},-d_2^{2/3},d_1^3,\rho_{12,3},\rho_{14,3},\rho_{24,3})\\ &+ K\cdot\exp(-r(T-t))N_3(-d_2^1,-d_2^2,-d_2^3,\rho_{12},\rho_{13},\rho_{23})\end{aligned} \tag{3.11}$$

3.4.2.2 Call on Max and Call on Min

From the price of the best of asset or cash option can the call max option price be derived. Note that:

$$\max\{\max\{S_1,S_2,S_3\}-K,0\}=\max\{\max\{S_1,S_2,S_3\},K\}-K=\max\{S_1,S_2,S_3,K\}-K$$

The call on max option is then a best of asset or cash option subtracted the strike. Realizing that fact it is easy to price the call on max:

$$\begin{aligned}\Pi_{cmax}(t,\mathcal{X}) &= S_1(t)N_3(-d_2^{2/1},-d_2^{3/1},d_1^1,\rho_{23,1},\rho_{24,1},\rho_{34,1})\\ &+ S_2(t)N_3(-d_2^{1/2},-d_2^{3/2},d_1^2,\rho_{13,2},\rho_{14,2},\rho_{34,2})\\ &+ S_3(t)N_3(-d_2^{1/3},-d_2^{2/3},d_1^3,\rho_{12,3},\rho_{14,3},\rho_{24,3})\\ &- K\exp(-r(T-t))\cdot\Big(1-N_3(-d_2^1,-d_2^2,-d_2^3,\rho_{12},\rho_{13},\rho_{23})\Big)\end{aligned} \tag{3.12}$$

Similar ideas are needed to derive the call on min, which we will not provide[7]. The fair price for the call on min derivative is:

$$\begin{aligned}\Pi_{cmin}(t,\mathcal{X}) &= S_1(t)N_3(d_2^{2/1},d_2^{3/1},d_1^1,\rho_{23,1},-\rho_{24,1},-\rho_{34,1})\\ &+ S_2(t)N_3(d_2^{1/2},d_2^{3/2},d_1^2,\rho_{13,2},-\rho_{14,2},-\rho_{34,2})\\ &+ S_3(t)N_3(d_2^{1/3},d_2^{2/3},d_1^3,\rho_{12,3},-\rho_{14,3},-\rho_{24,3})\\ &- K\exp(-r(T-t))\cdot N_3(d_2^1,d_2^2,d_2^3,\rho_{12},\rho_{13},\rho_{23})\end{aligned}$$

To derive the put of the exotic European options we can utilize a put-call-parity (p. 6 (Ouwehand, 2006)), but it takes a different form than the one presented in chapter 2 proposition 2.17. The put-call-parity for the exotic European call options is:

$$V_c(K)+K\exp(-r\cdot(T-t))=V_p(K)+V_c(0)$$

Where $V_c(K)$ is the value of the exotic European call option.

The exotic European options serve as an benchmark for the multivariate lattice approach, because if the lattice approach is close to the European benchmark then it is reasonable to expect a close estimate to the American option. The above section is also a presentation of the martingale theory versatility.

[7]The interested reader can read more on p. 6 in (Ouwehand, 2006)

Chapter 4

Deep Learning

The chapter is inspired by (Goodfellow, Bengio, and Courville, 2016; MacKay, 2018) where the interested reader can find more information about deep learning. Deep learning experiences a renaissance, because of the technology improvements in hardware and software. The collection of data has also significantly improved the field. Deep learning is a specialized field in machine learning, where you focus on a special architecture of models. Like in machine learning the basic components of a deep learning algorithm are a data set, cost function, optimization algorithm, and a model. E.g. in the LSM method we assumed the linear model, data set was the simulated paths, the cost function was the mean squared error and the optimization algorithm was a closed form solution of the normal equations. Deep learning is about studying neural networks which allows for greater flexibility than standard methods like linear regression.

A neural network consists of multiple layers, where the depth tells you how many layers the network has (figure 4.2), hence, the name "Deep learning". All the algorithms applied will be within supervised learning, where we try to fit the best relationship between the features and the response variable. Furthermore, all our algorithms will be based on the multilayer perceptron (MLP) model for regression, therefore, most of the chapter presents the theory for supervised MLP regression. The advantage of a multilayer model is that for each layer the updated set of covariates can be more finely tuned to better explain the data making the model extremely flexible. The MLP is also called a feedforward neural network, because the information only travels forward in the neural network, through the input node(s) then through the hidden layer(s) and finally through the output node(s). First, we present the basics for machine learning and then specialize the theory to deep learning.

4.1 Machine Learning Basics

The task for machine learning is to learn from data with respect to some performance measure. "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" ((Goodfellow, Bengio, and Courville, 2016) p. 97). Classical tasks T could be classification or regression, where the two methods differ on the output type. The former has discrete outputs, where regression has continuous output. We seek to find a price, which is naturally represented as a continuous value. Hence, our task will be to regress the price or expected continuation value. Measuring performance depends on the task, but for regression a

typical performance measure is mean squared error (MSE):

$$\frac{1}{K}\sum_{k=1}^{K}(y_k - \hat{y}_k)^2$$

There are other methods for measuring performance, e.g. mean absolute error (MAE). Another measure to quantify the fit is coefficient of determination:

$$R^2 = 1 - \frac{\frac{1}{K}\sum_{k=1}^{K}(y_k - \hat{y}_k)^2}{\frac{1}{K}\sum_{k=1}^{K}(y_k - \bar{y})^2} \quad \text{where } \bar{y} \text{ is the sample mean}$$

Coefficient of determination explains how well the model explains the data compared to the empirical mean. Some care should be taken when comparing models with different capacities, because the coefficient of determination will tend to prefer the larger models. The experience comes from data, where the data can be given with or without target values $\boldsymbol{y}$. In machine learning the task is quite different depending on if targets are given or not. Hence, the algorithms are split into two categories; supervised and unsupervised learning. The terminology *"supervised"* comes from a teacher that gives you the target values $\boldsymbol{y}$ that the algorithms tries to predict from $\mathbf{X}$.

Machine learning is not about making overly complex models to fit your data perfectly, because then the model will most likely have poor performance on unseen data. This phenomenon is known as overfitting, but the models can also be too simple, known as underfitting. If machine learning only cared about performance on the given data for training, then machine learning would be essentially optimization. The key difference is that machine learning wishes to obtain statistical generalization to unseen data. The practical way to evaluate the model is to measure generalization error on a test set. In machine learning the data is split into test data and training data, where the test set can only be used for evaluation after training.

The training data is used for training the model, where the training error shows how the fitted parameters fit to the training data. An unacceptable high training error could indicate that the regression method could be too simple and underfitting is the issue. For training it can sometimes also be useful to have a validation data set to see how your model generalizes, because the test set cannot be used to make model choices. It is common to split the training set into a validation and training set. The validation set is allowed to be used in training and it tries to mimic the generalization error. A common technique for measuring validation errors is *"cross-validation"*, where the data set is split into training and validation sets depending on the cross-validation scheme. The aim for the validation data is to approximate the model performance on unseen data, which is essentially what we want to have within an acceptable range.

After training the model we can evaluate the performance on a test set not seen in training. An unacceptable high test error when evaluating the model and very low training error could be an example of overfitting the model. Hence, when training the model, we want the best of both worlds; an acceptable training error, as well as making the gap between training and test error acceptable as well. In machine learning the overfitting and underfitting concepts are expressed by the relation of a

bias-variance trade-off, where the trade-off is between training error and generalization error. In case of overfitting, the model has been extensively trained to get a low biases, but the trade-off is that the model does not generalize well represented by a high variance.

A technique known as regularization is one approach to avoid overfitting. The regularization term is added to the cost function $J(\theta)$, which is essentially the function to minimize in order to train the model. There are many different regularization methods, where in section 4.2 we will present some useful methods for deep learning models. Besides the parameters estimated within the model, there are many parameters called hyperparameters exogenous given by the model designer.

Hyperparameters are the parameters not estimated within the model. Some common examples of hyperparameters in machine learning and deep learning are the learning rate, batch size, weight decay, model capacity, etc. The choice of the hyperparameters is often more important than the optimization algorithm chosen to estimate the parameters within the model. Hence, the need for suitable choices for hyperparameters is required for having a high quality machine learning model. The calibration of the model to a specific task is called hyperparameter tuning, where it can be done manually or automatically. The manual choice requires expert knowledge to set the hyperparameter optimal for the given task, where the automatic procedures are often computationally expensive. Examples of hyperparameters tuning rutines are random search, grid search, and Bayesian optimization.

The parameters within the model are found by minimizing the cost function $J(\theta)$. In some cases there exists either a closed form solution or the cost function is convex, making the optimization problem easier to handle. The complexity of MLP makes the cost function non-convex and iterative optimization procedures are needed. The basic iterative procedure is the gradient descent where the concept is to repeatedly making small moves in parameters toward better configuration. The most popular gradient methods in deep learning are Adam, RMSProp, AdaGrad, and stochastic gradient descent (SGD). The methods will be discussed in more details in section 4.2.3. To sum up, a machine learning model needs a data set, a model, a cost function, and an optimization algorithm. Understanding of basic machine learning will be the foundation for deep learning.

4.2 Multilayer Perceptrons

The goal of the multilayer perceptrons (MLP) is to approximate a function $f^*(x)$, where the MLP defines a mapping $x \in \mathbb{R}^R \mapsto f(x;\theta)$ to approximate $f^*(x)$. The task is to find the best θ such that the approximation $f(x;\theta)$ is close to the targets measured by a defined cost function $J(\theta)$. With the minimized cost function we seek to archive statistical generalization, i.e. good results for test data not used for training.

The first step for MLP is building the network, where we begin with zooming in on a single neuron, which is one node of the directed acyclic graph (figure 4.2).

4.2.1 A Single Neuron

The single neuron has R features x_r as inputs and one output $\hat{y}$ (figure 4.1).

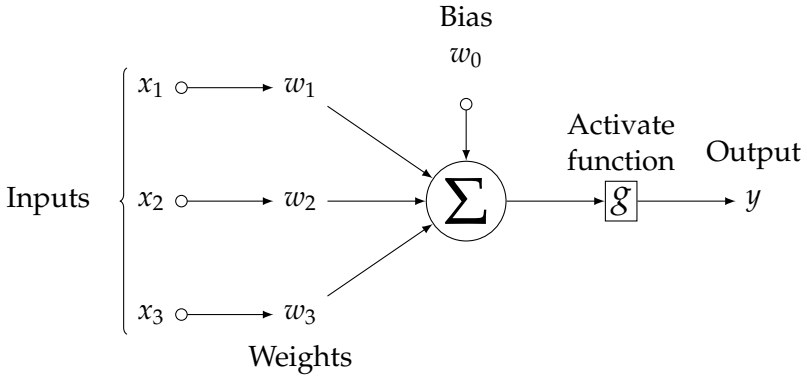


FIGURE 4.1: A Single Neuron

The function from inputs to output for a single neuron is:

$$\hat{y} = g(w_0 + \boldsymbol{x}^T \boldsymbol{w}) \quad where \quad \boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_R \end{pmatrix} \quad and \quad \boldsymbol{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_R \end{pmatrix}$$

The $\boldsymbol{w}$ is the weight matrix[1] and w_0 is the bias term. The term inside the function g is the activation of the neuron and it is a affine transformation denoted:

$$a = w_0 + \boldsymbol{x}^T \boldsymbol{w}$$

The function g is the activation function and it is essential for the flexibility of the MLP. There exists numerous of activation functions, only the imagination is the limit. We will list the most common and discuss them.

4.2.1.1 Activation Functions

Activation functions are essential for neural network, because they allow for non-linearities and flexibility. Activation functions apply a nonlinear transformation and decide whether a neuron should be activated or not. Without activation functions or the activation function being the identity function $g(a) = a$, the whole network would essentially be a linear regression model. Some popular activation functions are:

- Sigmoid function: $g(a) = \frac{1}{1+\exp(-a)}$

  This is the traditional choice, also called the logistic function. Popular in classification, but can suffer from vanishing gradient in deep learning.

- Hyperbolic tangent function: $g(a) = \frac{\exp(a)-\exp(-a)}{\exp(a)+\exp(-a)}$

  A scaled and shifted sigmoid function and likewise, suffers from the vanishing gradient problem. The range is $(-1, 1)$ and centered at zero. Often used for hidden layers.

[1] For a single neuron the weight matrix is a vector

- ReLU function: $g(a) = \max(0, a)$.

  Rectified Linear Unit is one of the most popular choices since it does not suffer from the vanishing gradient problem. The MLP often becomes more sparse because it sets some features to zero. It is claimed that ReLU learns multiple times faster than both the hyberbolic tangent and the sigmoid function.

- Leaky ReLU function:

$$g(a) = \begin{cases} a & if\ a \geq 0 \\ \alpha \cdot a & otherwise \end{cases}$$

  A disadvantage of ReLU is that some neurons may die out if the neurons are mapped to zero. The Leaky ReLU is designed to give such neurons a chance to get back into action, but not too easily, so $\alpha > 0$ is chosen small (typical $\alpha = 0.01$)

- ELU - exponential linear unit:

$$g(a) = \begin{cases} a & if\ a \geq 0 \\ \alpha(\exp(a) - 1) & otherwise \end{cases}$$

  Like the Leaky ReLU the ELU is designed to avoid the dead ReLU problem and the α is often chosen to be 1.

With a understanding of a single neuron we continue to the architecture of a MLP.

4.2.2 Architecture of MLP

A MLP consists of a input layer, where all R features enter, L hidden layers, and an output layer. Each hidden layer and the output layer consists of multiple neurons, where the width of the layer is the number of neurons in that layer (m^l) (figure 4.2). The network's inputs are called the input layer, the output layer is the output of the neural network. The layers between the input and output layer are hidden layers. This could be an explanation of why the field is called Deep learning, because of a deep structure of layers. In each hidden layer a linear combination of the features from the previous layer is made and an activation function is then applied in order to create the new hidden features in that layer (figure 4.2). The MLP is essential a large nested function, where the inputs go through a chain of functions until reaching the output.

$$\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{f}_1 \circ \boldsymbol{f}_2 \circ \cdots \circ \boldsymbol{f}_{L+1}$$
$$where\ \boldsymbol{f}_i : \mathbb{R}^{m^{i-1}} \to \mathbb{R}^{m^i} \quad i = 1, \ldots, L+1$$

Each function in the composition of functions corresponds to a layer of neurons.

$$\boldsymbol{f}_i(x) = \boldsymbol{g}(\mathbf{W}^T \boldsymbol{x} + \boldsymbol{w_0}) \quad x \in \mathbb{R}^{m^{i-1}},\ \mathbf{W} \in \mathbb{R}^{\boldsymbol{m^{i-1}} \times \boldsymbol{m^i}},\ and\ \boldsymbol{w}_0 \in \mathbb{R}^{\boldsymbol{m^i}}$$

So the function maps a vector to a vector, the hidden layers will often be denoted $\boldsymbol{h}$ where for each single neuron, we map a vector to a scalar by:

$$h_i = g(\boldsymbol{x}^T \cdot \mathbf{W}_{:,i} + (w_0)_i)$$

A layer is a vector of neurons, hence, the transformation from one layer to the next can be interpreted as multiple vector to scalar transformations, where each neuron acts in parallel. The different view motivates the initial presentation of single neuron network (section 4.2.1).

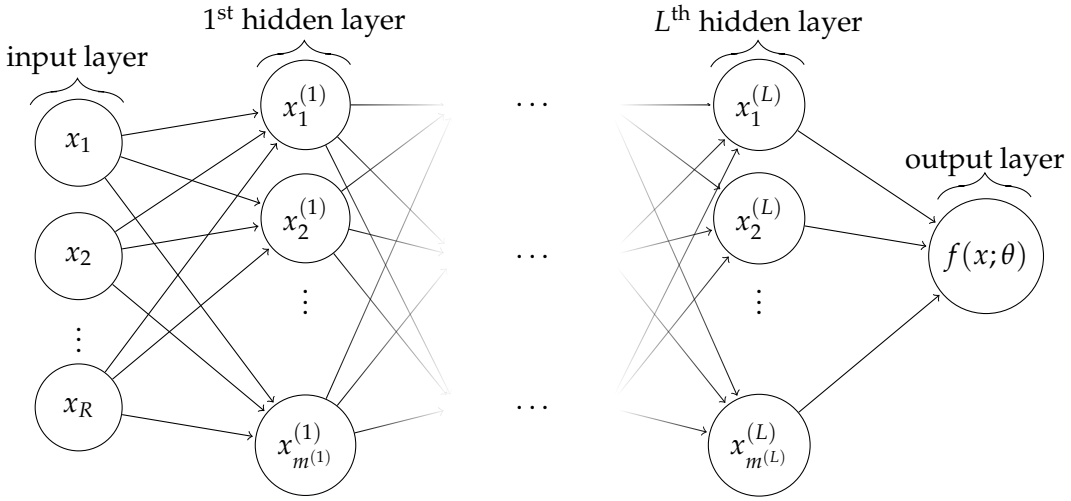


FIGURE 4.2: Multilayer perceptrons with $(L+1)$-layers with R input features and 1 output. The l^{th} hidden layer contains $m^{(l)}$ hidden neurons

So the MLP is not like the linear model in section 3.3, where a single transformation from input to output is applied. The unique attribute of neural network is the ability to approximate any kind of function[2], because of the flexibility with applying multiple functions to the input layer. The neural network has a lot of different design options, where e.g. hidden layers, layer width, depth, activation functions etc., are hyperparameters. In general, it is recommended to use many hidden covariates (neurons) rather than too few. The danger of overfitting is then avoided by introducing some kind of penalty to avoid that the model becomes overly complex, which we will discuss further in section 4.2.4.

To measure the performance of the model, we need a function to measure the difference between the approximation $f(\boldsymbol{x};\boldsymbol{\theta})$ and the target values $f^*(\boldsymbol{x})$. The function used to quantify this approximation is the cost function. The cost function is key to improving our model, in machine learning lingo training the model, which will be elucidated in the next section.

4.2.3 Training the Network

Training the network is key for building a high quality model. The performance of the model is measured by the cost function, where the cost function used in this

[2]Universal Approximate Theorem page 194 (Goodfellow, Bengio, and Courville, 2016)

thesis is the the empirical risk function.

$$J(\theta) = E_{(\boldsymbol{x},y)\sim\hat{p}_{data}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y) = \frac{1}{K}\sum_{k=1}^{K} L(f(x_k;\theta), y_k)$$

The loss function L in the empirical risk function for training is chosen to be quadratic, hence, for training we measure the error with mean squared error (MSE) as our cost function.

From the above, we see that the cost function is a way to measure the approximation of $f^*(x)$ by our model $f(\boldsymbol{x};\boldsymbol{\theta})$. The training is important, e.g., imagine after random initialization of parameter and construction of the MLP we reported the output given the inputs of the model. This model would probably results in a high cost function value, because the given weights would produce a function that does not fit training data. The way out of the high cost function is to minimize the cost function over the weightspace. Therefore, training for MLP is essentially a optimization of a non-convex function. Remember the MLP is a chain of functions (section 4.2.2) where the structure often makes the optimization a non-convex problem. Hence, a global minimum is seldomly archived. Other pitfalls, for optimization of MLP, are weight symmetry, steep cliffs, saddle points, vanishing gradients, and exploding gradients.

The actual optimization algorithms for MLP are based on gradients, where we make small local moves. The overall goal is to find the θ to achieve lowest possible test error. Within gradient methods there are batch gradient descend and mini-batch stochastic gradient descend, where the former is training on the whole data set, and the latter is only for a subset of the data set. The mini-batch methods have the advantage that the method can be parallelized for faster training. An epoch is the number of complete data set training cycles to update the weights. For the mini-batching techniques it is important to randomly sample from the whole data set in order to get unbiased gradient estimation.

The goal of the optimization algorithms is to find critical points $\nabla J(\theta) = 0$, hence, to obtain the critical points the iterative methods move in the opposite direction of sign of derivative $\nabla J(\theta)$. I.e. the gradient tells us how to update the parameters with a step size called the learning rate η:

$$\theta_{new} = \theta_{old} - \eta \nabla J(\theta_{old})$$

$J(\theta)$ is the cost function, hence remember we use the MSE:

$$J(\theta) = \frac{1}{K}\sum_{k=1}^{K} L(f(x_k;\theta), y_k) = \frac{1}{K}\sum_{k=1}^{K} (y_k - \hat{y}_k)^2$$

Common methods to estimate the parameters are gradient descent, stochastic gradient descent (SGD) and Adam, where all the optimization algorithms are iterative. Gradient descend uses the whole batch for each update, where Adam and SGD use mini-batches for each update. The Adam algorithm uses a adaptive learning rate, where the two others use constant or descending learning rates. The Adam method has greater progress in more gently sloped directions of weightspace compared to SGD and gradient descent.

Apart from choosing a optimization procedure, the initialization of the parameters are important. There are many suggestions to initialize parameters, but there is no general golden rule at the moment, because of a lack of understanding the optimization procedure in neural networks. Often, the practitioners tend to use simple and heuristic methods, where it has been shown that the initialization breaks weight symmetry.

The most common way of finding gradients is the backpropagation algorithm, where the basic idea is the chain rule from calculus.

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

To understand backpropagation it is often useful with a computational graph created by forward propagation, where the backpropagation computes the derivative from output layer to input layer, by going backwards in the computational graph. The training process is a forward-backward algorithm. Different starting values of θ will result in different parameters. The good news is that these predictors typically do not differ by very much. It is recommended to work with a set of different starting values and then use as a final predictor the average of the individual predictors originating from each starting value.

4.2.4 Regularization

The number of parameters and the capacity of neural networks often resilt in overfitting, i.e. the model does not generalize well on new data. Regularization is a technique that alters our optimization problem to discourage complex models, hence, avoiding the problem of overfitting. Some common methods for deep learning are parameter norm regularization, early stopping, and dropout.

Early stopping is an effective and simple method for regularization. Compared to parameter norm regularization, the early stopping algorithm does not harm the learning dynamics. The early stopping method is also computationally efficient, and therefore, it is a popular regularization method for deep learning. The idea of early stopping is that the iterative training algorithm keeps improving the train error, but training too extensively leads the the gap between training and test error to increase. The idea is then to split your data into validation and train data sets, where you determine the best $\hat{\theta}$ and the corresponding training steps $\hat{i}$ by iterative comparing the cost function on the validation set. The algorithm stops after a predefined number of steps without improving the cost function on the validation set. The number of epochs to wait for the cost function not to improve is called the patient for the early stopping algorithm.

Like early stopping the dropout method is computationally inexpensive. The idea is to remove neurons randomly at each training loop, i.e. when updating the gradient, each node is kept with probability p, independently of each other. To perform dropout, a binary mask is sampled independently for each iterations, where the probability p for 1 is another hyperparameter. The goal is to minimize $E_{\boldsymbol{\mu}} J(\boldsymbol{\theta}, \boldsymbol{\mu})$ where $\boldsymbol{\mu}$ is the mask. The results of the procedure are more robust features and a regularizing effect on most models.

There are many design options for deep learning, where the choices should be specific for the given task. There is a no free-lunch theorem for machine learning, which says no model is superior for all tasks (page 114 (Goodfellow, Bengio, and Courville, 2016)). The designs for specific tasks are important where training error and test error can be improved by designing the model for the given task. Another aspect is the computational resources, i.e. the memory space and computational time.

Pricing derivatives with deep learning methods have two clear benefits. The first is computational time, where, after a model is trained, then the model is superior to the methods presented in chapter 3. Another advantage is the nonlinearities of the model, making it possible to fit more complex functions. The application of deep learning in option pricing theory will be explored in the next chapters.

Chapter 5

Option Pricing and Deep Learning

Deep learning can be applied to option pricing theory in different ways. We will investigate two methods using MLP regression. The first method considered is a modified LSM algorithm, where MLP regression is used to estimate the expected contuation value instead of the linear model. The second method uses MLP regression on a dataset $(\mathbf{X}, \boldsymbol{y})$ to infer prices. The input parameters $\mathbf{X}$ are simulated and the target variable $\boldsymbol{y}$ is found with existing pricing methods.

The two methods will be compared numerically in the next chapter. Both methods fall within supervised regression where we will use MLP introduced in section 4.2 to approximate the mappings. The risky assets are modeled with Black-Scholes theory from earlier chapters, hence, the appropriate simulating methods are already presented (Chapter 2, 3). The theory in chapter 4 will be specialized for the specific task. The advantage of MLP is that the model scales well to high dimensional data, where, e.g. polynomial regression (section 3.3) is prone to overfitting and slow compared to MLP for high dimensional tasks.

5.1 Multilayer Perceptrons Regression for Optimal Stopping

The first application of neural networks is to investigate, if the LSM method can be improved by using MLP regression to approximate expected continuation value instead of using the linear model in LSM. This section is influenced by (Longstaff and Schwartz, 2001; Lapeyre and Lelong, 2019; Kohler, Krzyżak, and Todorovic, 2010), where the approximate scheme with only in-the-money paths (ITM) is inspired by (Longstaff and Schwartz, 2001) and the idea of using neural network instead of the linear model comes from (Kohler, Krzyżak, and Todorovic, 2010; Lapeyre and Lelong, 2019). The algorithm is as usual based on approximating the American put option with a Bermudan option.

The setup is the same as for the LSM presented in chapter 3, where the difference is the regression to estimate the expected continuation value. The main difference of the two methods lies in the capacity of the two regression models. The MLP through its chain of functions and activation functions can capture complex structures. Another advantage is that deep learning is known for breaking the curse of dimensionality, hence, the MLP could be more suited for multivariate contingent claims than the classical LSM.

5.1.1 Recap MLP

We model the MLP with a nonlinear function:

$$s \in S \in \mathbb{R}^R \mapsto \Psi(s;\theta) \in \mathbb{R}$$

where Ψ is the function decomposition and given by:

$$\Psi = a_{L+1} \circ g_L \circ a_L \circ \cdots \circ g_1 \circ a_1 \quad \textit{where } L \in \mathbb{N}$$

We refer to chapter 4 for notation, where $a_l(\boldsymbol{x}) = \mathbf{W}_{(l)}^T \boldsymbol{x}^{(l-1)} + \boldsymbol{w}_0^{(l)}$. We collect all the parameters of the different layers into a high dimensional parameter:

$$\theta = (\mathbf{W}_l, \boldsymbol{w}_{0,l})_{l=1,\ldots,L+1} \in \mathbb{R}^{N_m} \text{ with } N_m = \sum_{l=1}^{L+1} m^{(l)}(1 + m^{(l-1)})$$

We will restrict our parameter space by a increasing sequence $(\gamma_p)_{p\in\mathbb{N}}$ such that $\lim_{p\to\infty} \gamma_p = \infty$ and $p \in \mathbb{N}$ is the maximum number of neurons in each hidden layer. We define the set:

$$\Theta_p = \{\theta \in \mathbb{R} \times \mathbb{R}^p \times (\mathbb{R}^p \times \mathbb{R}^{p\times p})^{L-1} \times \mathbb{R}^p \times \mathbb{R}^{p\times p} : |\theta| \leq \gamma_p\}$$

By the definition of the set, we restrict our MLP to the set:

$$\mathcal{NN}_p = \{\Psi(\cdot;\theta) : \theta \in \Theta_p\}$$

Unfortunately the $\mathcal{NN}_p$ is not a vector space, hence, the regression cannot approximate the expected continuation value with a projection onto a finite vector space as for the LSM. The MLP method is justified by the "Universal Approximate Theorem" (theorem 5.1)

Theorem 5.1. ***Universal Approximation Theorem*** *Assume that the activation function g is non constant and bounded. Let μ denote the probability measure on $\mathbb{R}^r$, then for any $L \geq 1$, then $\mathcal{NN}_\infty$ is dense in $L^2(\mathbb{R}^r, \mu)$*
where $\mathcal{NN}_\infty = \cup_{p\in\mathbb{N}} \mathcal{NN}_\infty$

(p. 4 (Lapeyre and Lelong, 2019))

Note that the above theorem can be rephrased in terms of approximating random variables.

Remark 5.2. *Let Y be a real valued random variable such that $E[Y^2] < \infty$. Let X be a random variable taking values in R^r and $\mathcal{G}$ be the smallest σ-algebra such that X is $\mathcal{G}$ measurable. Then, there exists a sequence $(\theta_p)_{p\geq 2} \in \prod_{p=2}^{\infty} \Theta_p$ such that $E[|Y - \Psi_p(X;\theta_p)|^2] = 0$. Therefore, if for every $p \geq 2$, $\alpha_p \in \Theta_p$ solves*

$$\inf_{\theta\in\Theta_p} E[|\Psi_p(X;\theta) - Y|^2]$$

Then the sequence $((\Psi_p(X;\alpha_p))_{p\geq 2})$ converge to $E[Y|X]$ in $L^2(\Omega)$ when $p \to \infty$

(p. 5 (Lapeyre and Lelong, 2019)).

The remark reveals that the MLP can be used to approximate the conditional expectation instead of the linear model in LSM.

5.1.2 The Algorithm

The algorithm is similar to the LSM, but the regression step is slightly different. We will use the same assumptions as in LSM section and same principles. Recall that the optimal stopping problem can be solved with dynamic programming principle on the optimal policy:

$$\begin{cases} \hat{\tau}_{t_N}^{k,p,K} = t_N \\ \hat{\tau}_{t_n}^{k,p,K} = t_n \cdot 1_{\{G(S^{(k)}(t_n)) \geq \Psi_p(S^{(k)}(t_n);\hat{\theta}_{t_n}^{p,K})\}} + \hat{\tau}_{t_{n+1}}^{k,p,K} \cdot 1_{\{G(S^{(k)}(t_n)) \geq \Psi_p(S^{(k)}(t_n);\hat{\theta}_{t_n}^{p,K})\}} \end{cases} \quad for\ n = 0, \ldots, N-1$$

Where the estimator $\hat{\theta}_{t_n}^{p,K}$ is given by minimizing squared sample distance of the estimated continuation value and the realized continuation value:

$$\hat{\theta}_{t_n}^{p,K} = \underset{\theta \in \Theta_p}{\operatorname{argmin}} \sum_{k=1}^{K} 1_{\{G(S^{(k)}(t_n))>0\}} \left(G(S^{(k)}(\tau_{t_{n+1}}^{k,p,K})) - \Psi_p(S^{(k)}(t_n);\theta) \right)^2$$

Finally, in analog with the LSM section the approximated time 0 price for the option is:

$$U^{p,K}(0) = \max\{G(S(0)), \frac{1}{K} \sum_{k=1}^{K} G(S^{(k)}(\hat{\tau}_{t_1}^{k,p,K}))]\}$$

5.2 Multilayer Perceptrons Regression Pricing from Existing Methods

The MLP pricing method from existing methods has a different approach than the other methods, because it is only data driven. The MLP could easily be used on real data, which is investigated in (Gaspar, Lopes, and Sequeira, 2020). We revisit the work from (Hirsa, Karatas, and Oskoui, 2019), where we try to extend the pricing method to options with two underlying risky assets. The model will be the classical Black-Scholes model presented in earlier chapters. By choosing this model the MLP pricing method is ready for investigation both for vanilla options and exotic options. We stress that the MLP pricing method is not restricted to the Black-Scholes model, but can be applied to other models such as Hesten, Variance Gamma, and real market data. The advantage of MLP is the fast pricing once trained. With the increased speed for pricing it can cost accuracy, especially if the data is sparse, which can arise when using the method for exotic options on real market data. For practical application the accuracy is severe if the predicted price leads to arbitrage. We consider the method for European call, American put, and American put miminum on two asset options presented in earlier chapters.

For deep learning the hyperparameters are important for finding the right model for pricing, where different choices will be presented empirically under training. For the given task polynomial regression can also be used, but we see later why MLP regression is preferred. The section is split into three sections "Data", "Training", and "Performance".

5.2.1 Data

The generation of labels is the computational expensive part of the MLP method, because the method needs enough samples to approximate the function f^* well. The upside after generation of labels is that the method is computationally fast and easy to implement with basic knowledge of deep learning seen in chapter 4. The labels will be generated by existing methods presented in chapter 2 and 3, where the input parameters will be sampled from a uniform distribution or quasi sampled with Halton sequences.

For both European and the American univariate contingent claims the parameter in-sample will be the same, where we remember the 5 parameters for pricing the univariate contingent claims[1]. Note for simulation of labels the European call and American put options are first order homogeneous function in $(S(0), K)$, hence, the valuation formulas can be modified:

$$\frac{c(S(0),K)}{K} = c(\frac{S(0)}{K}, 1) \quad \textit{and} \quad \frac{P(S(0),K)}{K} = P(\frac{S(0)}{K}, 1)$$

The alternative representation above reduces the number of parameters needed for simulation to 4, because instead of simulate both S and K, only moneyness $(\frac{S(0)}{K})$ is simulated. This property is also shared for the American bivariate contingent claim. For the European call and American put the input matrix $\mathbf{X}$ is different combinations of the 4 parameters within the parameter range (table 5.1), where it is assumed the number of trading days are 252 in a year. The parameter ranges are risk-free rate of

[1] E.g. the European call proposition 2.16

return between 1-3 %, maturity between 1 day to 3 years, moneyness between 0.8 and 1.2 and volatility between 0.05 and 0.5.

TABLE 5.1: Parameter ranges for American put and European call options

Derivative	**r**	**T**	**Moneyness**	σ
Euro. Call	1%-3%	1d - 3y	0.8-1.2	0.05-0.5
Amer. Put	1%-3%	1d - 3y	0.8-1.2	0.05-0.5

The American put minimum option for two underlying assets will require additional parameters, because now we have two spots, two volatilities, and correlation between the assets. The first order homogeneity does also hold for bivariate contingent claims.

$$\frac{P(S_1(0), S_2(0), K)}{K} = P(\frac{S_1(0)}{K}, \frac{S_2(0)}{K}, 1)$$

The parameters considered for the bivariate contingent claims are T, r, $Moneyness_1$, $Moneyness_2$, σ_1, σ_2 and ρ. So a total of 7 parameters and the given parameters ranges are given in table 5.2.

TABLE 5.2: Parameter ranges for American put minimum on two assets

r	**T**	$Moneyness_1$	$Moneyness_2$	σ_1	σ_2	ρ
1%-3%	1d - 3y	0.8-1.2	0.8-1.2	0.05-0.5	0.05-0.5	(-0.5)-0.5

The simulation of the parameter ranges for the training data set is done by quasi-random Halton sequences sampling to obtain lower discrepancy, where the test sets are sampled with random uniform sampling. The quasi random sampling is different from uniform random sampling, since the purpose is not to mimic randomness. Using Halton sequences sampling, the aim is to increase accuracy by generating points that are too evenly distributed to be random. The Halton sequence and uniform sampling generates points between 0 and 1, hence, we need to apply a transformation to have the parameter ranges:

$$Simulated\ point \cdot (range\ of\ parameter) + lower\ bound\ of\ parameter$$

After the simulation of the input parameters within the given ranges, the labels can be generated from existing methods. For the European call option the generation of labels is done by the classical B-S formula for call options given in proposition 2.16. The B-S pricing formula is well-known and has an analytical solution, hence, it is relatively fast to generate labels in this model. The American options require numerical methods, therefore, the generation of labels is more computationally expensive. The labels for the American options are generated by CRR with 100 equidistant time-steps for the American put option with one underlying stock and by BEG for two underlying assets with 50 equidistant time-steps presented in chapter 3. When the data sets ($\mathbf{X}$, $\boldsymbol{y}$) are generated, we are left with a regression task. The regression task

is to predict the price $\boldsymbol{y}$ from the input parameters $\mathbf{X}$.

The data sets generated for within the parameter ranges will be referred to as the in-sample data set. The training data set is an in-sample data set, which is split up into a training and validation data set in order to approximate model performance on the test sets. The training data set is used to update the parameters internally in the model, e.g. weights, biases, etc., and to fine-tune hyperparameters for choosing the optimal model design. To avoid overfitting and have good generalization models the validation set is useful, because the trained model has not seen the data before evaluation on the validation set. The validation set is randomly subsampled from the training set, where the validation data set constitutes 20 percent of the training set. To check the robustness of the regression we choose to sample three test sets with one in-sample and two out-of-sample data sets.

The out-of-sample test data sets are simulated by uniform sampling adjusting the parameter range for either moneyness or maturity for the options (table 5.3). The test data set has not been seen by the model in the training process, hence, we get an unbiased evaluation of the model. The aim with producing different test data sets is to measure the models' performance at interpolation and extrapolation. The data set for the bivariate American contingent claim will be similar to the American put by adjusting either the $moneyness_1$ and $moneyness_2$ or the maturity.

TABLE 5.3: Parameter Ranges in Test Data Set for European Call and American Put Option

Parameter ranges for European call and American put for test data sets

Data set	Derivative	**r**	**T**	**Moneyness**	σ
In-Sample	Euro. Call	0.05-0.5	1d-3y	0.8-1.2	1%-3%
Out-Of-Money		0.05-0.5	1d-3y	0.6-0.8	1%-3%
Longer Maturity		0.05-0.5	3y-5y	0.8-1.2	1%-3%
In-Sample	Amer. Put	0.05-0.5	1d-3y	0.8-1.2	1%-3%
In-The-Money		0.05-0.5	1d-3y	0.6-0.8	1%-3%
Longer Maturity		0.05-0.5	3y-5y	0.8-1.2	1%-3%

In figure 5.1 we have visualized a simulated training data set for the American put. The marginal distributions shown is for 300.000 data samples $(\mathbf{X}, y)$ generated by Halton sequences and the CRR model. The marginal distributions for the features cover the parameter range almost uniformly and the simulated y lies with most values at zero and maximum at 0.387 rounded to three decimals. The marginal distributions show that we have successfully generated parameters in the given ranges and the parameters are evenly spaced in the ranges. In the model performance section the out-of-sample and in-sample test data sets will be used to check the extrapolation and interpolation of the models. The test data sets are 60.000 generated data points with uniform sampling.

5.2.2 Training

The aim of training is that the model will learn the pricing function f^*. Once the data set $(\mathbf{X}, \boldsymbol{y})$ is generated the model can be trained to approximate the true function f^*.

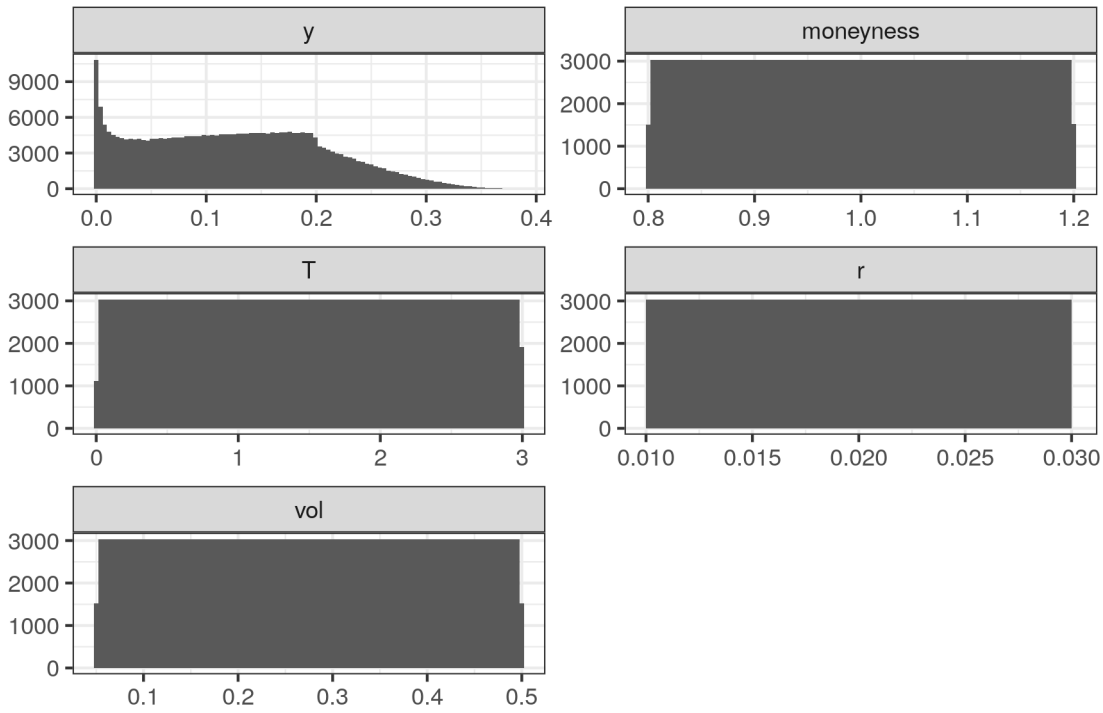


FIGURE 5.1: Quasi random simulation with Halton sequences for input variables and CRR for generation of labels for American put option.

We will also present a standard linear model to compare the MLP with. We want to have a fast, robust, and accurate model after training on the training set. To train the model we need a measure for the error, where the mean squared error (MSE) for regression is applied. I.e. the cost function is chosen to be the empirical risk function with a quadratic loss function:

$$J(\theta) = \frac{1}{K}\sum_{k=1}^{K}(y_k - \hat{y}_k)^2$$

The MSE penalizes outliers stronger than, e.g. mean absolute error (MAE), but it penalizes small deviations less. To avoid overfitting we regularize using early stopping with a patient of 5 epochs, but with a maximum of 100 epochs for computational reasons. To update the weights the Adam optimization algorithm is chosen. The learning rate η is found by hyperparameter tuning for the American put minimum on two assets option and chosen to be 0.001 for the univariate contingent claims.

5.2.2.1 Hyperparameter Tuning

A research area within deep learning is to fine-tune the hyperparameters to the specific task, where both manually and automated searches can be used. To choose the best hyperparameters there are several choices, where the most basic automated task is random search and grid search. The random search is to define a predefined

range to pick randomly from. This method can be effective to discover new hyperparameter values or combination, but it can be considerably more computationally expensive than the grid search. The grid search is used for searching in a grid with each hyperparameter in each dimension. The grid search has fast computational time relative to the random search, but requires some expert knowledge about the hyperparameters.

We test empirically the best hyperparameters for the MLP, where the validation set is used. For hyperparameter tuning grid search is conducted to find the optimal set of hyperparameters, where we will look at data set size, learning rate, and batch size.

Firstly, the European call option regression is investigated, where we vary the data set size and batch size. The goal is to quantify how large a data set(d) and batch size(b) are needed for having a high quality model for the European option. The data sets for an European call option considered are in-sample data sets of size 1K, 10K, 100K, 300k and 1M data samples, where the validation set is subsampled from the training data set. The batch sizes are 64, 256 and 1024, i.e. we have the combinations expressed by the Cartesian product:

$$b \times d = \{(b,d) : b \in \{64, 256, 1024\} \; and \; d \in \{1K, 10K, 100K, 300K, 1M\}\}$$

By inspiration from (Hirsa, Karatas, and Oskoui, 2019) we train a MLP model with 4 layers, 120 neurons in each hidden layer and 1 neuron in the output layer. In each layer we choose the activation function leaky ReLU with negative slope α=0.01, which is one of the most popular choices for activation functions. The number of data samples is relevant for real data, because for real market data there is not unlimited market data available.

Table 5.4 shows that the the model performs well for in-sample data with 10K-300K data points. The biggest data set with 1M seems not to be worth the computational cost. The model is only trained once on each data sets, so there is some randomness on each run. The model to interpolate prices for European call options in-sample data does not significantly improve with gathering more data than 10K-300K data points, which is good news for using the method on real market data. Beware that the simulated data can underestimate the validation error, because we have a controlled setup where the parameter range is within a predetermined range. For the controlled setup with simulated data we can choose arbitrary many data points, albeit making the method more computationally expensive. By weighting both the computational cost and accuracy, we choose to work with 300K data points and a batch size of 64 for the European option.

The European option needs fewer parameters compared to the American put on minimum of two assets, hence, the data set might need to be larger for that study. We conduct a large grid search with variations of the learning rate η, the batch size b and the data set size d, where the grid of the Cartesian product of η, b and d is searched.

$$\eta \times b \times d = \{(\eta, b, d) : \eta \in \{0.0001, 0.001, 0.01\}, \; b \in \{8, 64, 256, 512, 1024\} \; and \; d \in \{1K, 100K, 300K\}\}$$

Besides varying the hyperparameters for η, b and d, the other hyperparameters are

TABLE 5.4: Grid Search for European Call

Hyperparameter tuning of data set size and batch size for the European call option. The table shows validation loss in ascending order for different hyperparameter combinations and for the interested reader the tensorboard is online (link tensorboard 1)

Data set Size	Batch Size	Training Loss	Validation Loss	Time: min:sec
1M	256	$8.094 \cdot 10^{-7}$	$8.7764 \cdot 10^{-7}$	3:38
300K	64	$7.0562 \cdot 10^{-7}$	$1.0849 \cdot 10^{-6}$	2:49
1M	1024	$1.1596 \cdot 10^{-6}$	$1.1578 \cdot 10^{-6}$	5:58
300K	256	$9.579 \cdot 10^{-7}$	$1.3268 \cdot 10^{-6}$	2:29
300K	1024	$1.5367 \cdot 10^{-6}$	$1.4138 \cdot 10^{-6}$	9:28
1M	64	$3.4919 \cdot 10^{-7}$	$1.9197 \cdot 10^{-6}$	8:24
100K	256	$2.243 \cdot 10^{-6}$	$2.1192 \cdot 10^{-6}$	1:02
100K	64	$1.9565 \cdot 10^{-6}$	$2.5738 \cdot 10^{-6}$	1:01
100K	1024	$3.22 \cdot 10^{-6}$	$4.4754 \cdot 10^{-6}$	2:00
10K	256	$1.1179 \cdot 10^{-5}$	$1.0980 \cdot 10^{-5}$	0:37
10K	64	$1.0043 \cdot 10^{-5}$	$1.9830 \cdot 10^{-5}$	0:15
1K	64	$6.1389 \cdot 10^{-5}$	$7.8711 \cdot 10^{-5}$	0:22
10K	1024	$8.7067 \cdot 10^{-5}$	$8.1122 \cdot 10^{-5}$	0:32
1K	256	$1.2032 \cdot 10^{-4}$	$1.2504 \cdot 10^{-4}$	0:20
1K	1024	$7.5948 \cdot 10^{-3}$	$7.3595 \cdot 10^{-3}$	0:08

the same values as for the European call option.

Looking at table 5.5 the best configuration for the training loss is for $(\eta = 0.0001, b = 64, d = 300K)$, where the best configuration in terms of validation loss is $(\eta = 0.001, b = 64, d = 300K)$. The computational burden increases when training with a bigger data set, smaller learning rate, and smaller batch sizes. The learning rate effects the computational speed because we are training with early stopping regularization. The batch size effects also the computational speed, since a high batch size gives fewer iterations per epoch. Overall the data set with 300K performed best, which is expected, since a bigger data set gives more learning opportunities. Considering computational cost and the loss; the 300K data sets, the learning rate $\eta = 0.001$ and a batch size of 64 are preferred.

Hyperparameter tuning is highly computationally expensive and there are many combinations to test out. With grid search we have seen a possible optimal choice for the learning rate, batch size, and data set size for the European call option and American put minimum on two assets. The American put option uses the same parameters as the European option, therefore, we choose the same hyperparameters for the univariate contingent claims. Below, we will try a different model than MLP to see if we really need deep learning for this method.

5.2.2.2 Polynomial Regression

To compare MLP and Polynomial regression the data set and the performance metrics are the same, but the model training is obviously different. The chosen data set size is with 300.000 samples and the performance metric is MSE. For simplicity the

TABLE 5.5: Grid Search for American Put Minimum on two Stocks

Hyperparameter tuning of data set size, learning rate and batch size for the American put bivariate contingent claim. The table shows the top 10 best performing combinations for the training loss and for the interested reader the tensorboard is online (link tensorboard 2) and the full table is given in appendix table D.2

Data set Size	**Learning Rate**	**Batch Size**	**Train Loss**	**Val. Loss**	**Time: min:sec**
300K	0.0001	64	$1.350 \cdot 10^{-6}$	$2.287 \cdot 10^{-6}$	4:50
300K	0.001	64	$2.170 \cdot 10^{-6}$	$1.633 \cdot 10^{-6}$	2:13
300K	0.0001	8	$2.734 \cdot 10^{-6}$	$2.638 \cdot 10^{-6}$	6:17
300K	0.0001	256	$2.943 \cdot 10^{-6}$	$2.850 \cdot 10^{-6}$	4:04
300K	0.001	256	$3.685 \cdot 10^{-6}$	$3.335 \cdot 10^{-6}$	1:26
100K	0.0001	64	$4.081 \cdot 10^{-6}$	$3.575 \cdot 10^{-6}$	1:44
100K	0.001	64	$4.817 \cdot 10^{-6}$	$5.615 \cdot 10^{-6}$	0:57
100K	0.0001	256	$5.182 \cdot 10^{-6}$	$6.367 \cdot 10^{-6}$	1:52
300K	0.0001	1024	$5.534 \cdot 10^{-6}$	$6.473 \cdot 10^{-6}$	3:42
300K	0.001	8	$5.798 \cdot 10^{-6}$	$4.284 \cdot 10^{-6}$	5:40

European call option is investigated, where we fit polynomials up to degree 6 for comparison of the model capacity and fit.

$$y_i = \beta_0 + \beta_1 \cdot x_i + \cdots + \beta_n \cdot x_i^n + \epsilon_i \quad \textit{where } n = 1, 2, \ldots, 6$$

Plotting the fit actual vs predicted target variable (figure 5.2), it is clear that the in-sample fit improves with the increased model capacity. The linear regression is too simple for pricing European option, but the 6 order polynomial actually performs better than the MLP for the in-sample validation set (table 5.6). Keep in mind that we do not only want good interpolation, but we also want good extrapolation for our fitted model. The out-of-sample data will reveal if the high order polynomial or MLP have overfitted the data.

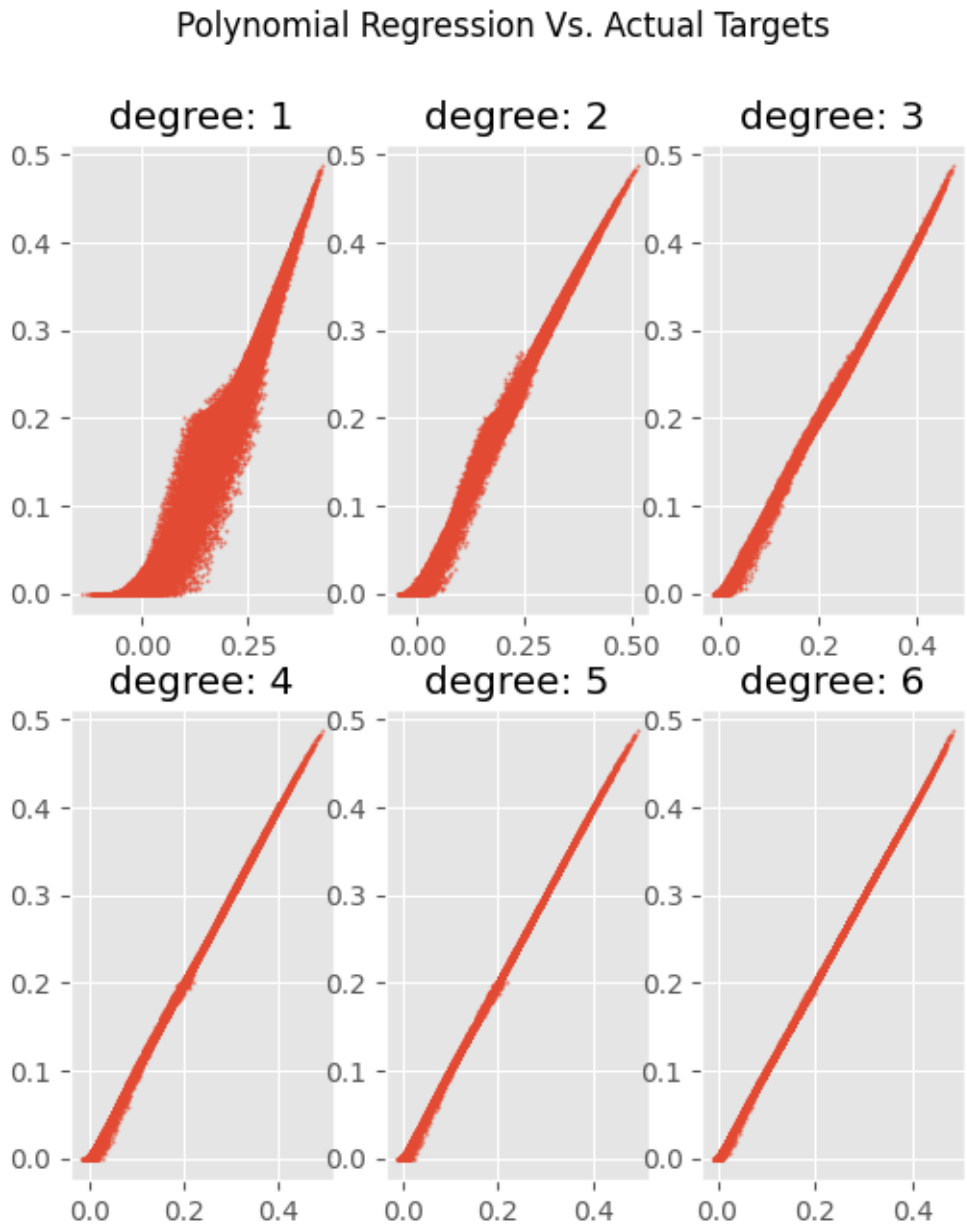


FIGURE 5.2: Predicted price based on polynomial regression of varying degree

Looking closer at table 5.6 to compare the performance of each model. The table confirms that the linear regression has a worse fit than the other models with higher capacity. The difference on the MLP and best performing polynomial model are less than $2 \cdot 10^{-6}$. The difference is almost negligible, so the fit for MLP and polynomial regression of degree 4-6 all perform well on the in-sample training data in terms of the validation loss.

TABLE 5.6: European Call Validation Error

Validation error of polynomial regression and MLP for the European call option

Model	**Valalidation Loss**
1. degree poly.	0.000631
2. degree poly.	0.000069
3. degree poly	0.000013
4. degree poly.	0.000004
5. degree poly.	0.000002
6. degree poly.	0.000001
MLP	0.000003

5.2.3 Performance

The model performance is evaluated by MSE, RMSE, MAE and coefficient of determination, where all the measures evaluate how close the model predictions are with the actual targets. The first three measure ranges are $\mathbb{R}^+$, where the goal is to have the lowest value possible. MSE close to 0 means that the model predictions do not differ a lot from the observed targets. The RMSE and MAE are the same type of measure, but the deviation is measured slightly different. The coefficient of determination has range $(-\infty, 1]$, where a higher value indicates a better model. Coefficient of determination provides a measure of how well observed targets are predicted by the model based on the proportion of total variation of targets explained by the model.

5.2.3.1 European Call Option

The European call option is trained with the algorithm and hyperparameters described in the training section (section 5.2.2). By the hyperparameter investigation we choose a batch size of 64 and a data set of 300K samples. We compare the MLP regression with the polynomial regression. Table 5.7 shows that the MLP is superior at extrapolating, because the MLP performs better on all metrics on the out-of-sample test data sets compared to fitted polynomial regressions with a degree between 1 and 6[2]. For the in-sample test data set the polynomial regression of order 6 and MLP perform almost equally well. Note, the in-sample test data is similar to the in-sample training data, hence, we expect the same magnitude of error for the test set as for the training set for in-sample testing. The performance measures show that the polynomial regression, that was performing well on the in-sample data set, was due to overfitting, because the high order polynomial regression performs poorly on out-of-sample data (table 5.7). For the 6. order polynomial regression, we see a negative coefficient of determination, which means the model performs worse than the model that predicts the sample mean.

[2]The out-of-sample fits for the European call with polynomial regression are illustated in figure **??** and figure **??**

TABLE 5.7: Test Error for European Call Option

Performance comparison of MLP and polynomial regression for the European call option. Shown the best performing regressions in the linear model and the worst performing in terms of MSE for in-sample and out-of-sample test data sets (The full table for polynomial regression is found in appendix table D.1)

Model	Data set	MSE	RMSE	MAE	Coefficent of Determination
MLP	In-Sample	0.000000	0.000629	0.000486	0.999961
	Out-of-Money	0.000007	0.002644	0.001551	0.995911
	Long Matuiry	0.000197	0.014048	0.010061	0.986518
6. degree poly.	In-Sample	0.000001	0.000958	0.000591	0.999909
1. degree poly.		0.000636	0.025212	0.018326	0.936628
2. degree poly.	Long Maturity	0.001196	0.034577	0.026287	0.918316
6. degree poly.		0.043361	0.208233	0.111190	-1.962442
2. degree poly.	Out-Of-Money	0.000767	0.027694	0.022203	0.551246
1. degree poly.		0.005772	0.075973	0.060936	-2.377251

The MLP has high predictive strength compared to the polynomials, because it performs well, also on out-of-sample data sets. The MLP show that it predicts out-of-money call options with less than a MSE value of 10^{-5}, where the fit for the long maturity test set has a higher MSE. The European options are liquid in the markets, hence, the pricing method can easily be trained on real data. The MLP fit for in-sample data on the European call option is visualized in figure 5.3, which shows $\frac{c(S_0,K)}{K}$ predicted from the model and observed target values $\boldsymbol{y}$. The figure shows a overall close fit. For the remaining part of this section only the MLP regression will be considered, because the polynomial regression has shown bad performance for out-of-sample data.

5.2.3.2 American Put Option

The American put option is priced with the same MLP algorithm as for the European call. Table 5.8 shows once again a good fit, hence, we believe we have a high quality model for the American put option[3].

TABLE 5.8: MLP Performance on American Put Option

Data set	MSE	RMSE	MAE	R^2
In-Sample	0.000002	0.001562	0.001278	0.999634
In-The-Money	0.000012	0.003519	0.002290	0.995778
Longer Maturity	0.000193	0.013894	0.009213	0.980835

5.2.3.3 American Put on Minimum of two Assets Option

The American put on minimum of two assets option has almost double the number of parameters compared to the univariate contingent claims. Hence, the performance might produce a slightly higher MSE. Table 5.9 and figure 5.4 show that the

[3]The out-of-sample fits for the American put with MLP regression are illustated in figure **??** and figure **??**

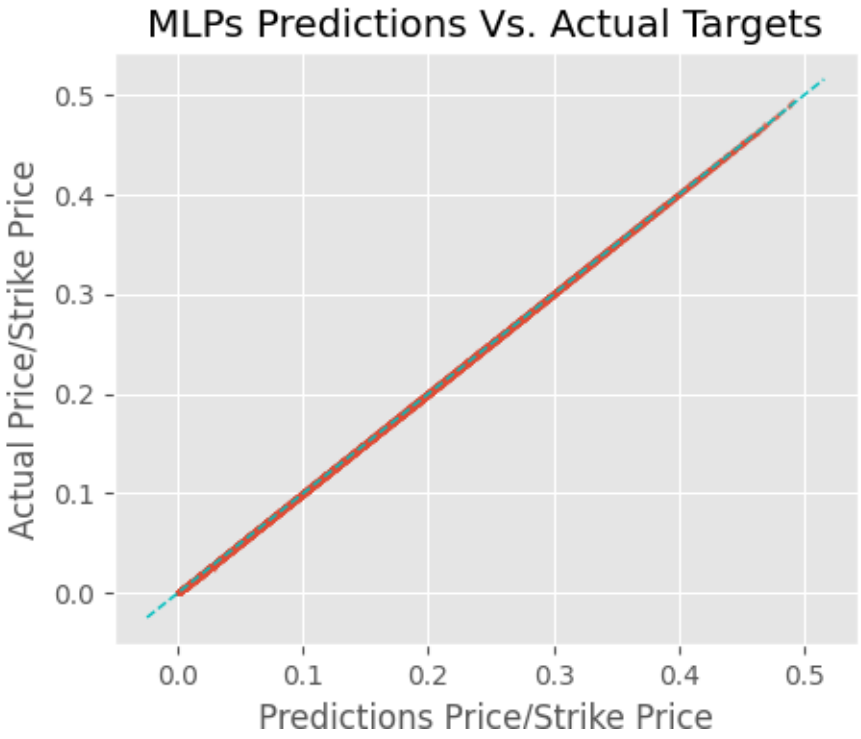


FIGURE 5.3: MLP Performance on in-sample dataset on European call

MLP also fits the American put minimum option on two assets well. The performance seems similar to the univariate case, hence we see that the MLP can easily fit bivariate contingent claims as well.

TABLE 5.9: Performance of the Bivariate American Put Contingent Claim

Data set	**MSE**	**RMSE**	**MAE**	R^2
In-Sample	0.000002	0.001361	0.001008	0.999794
In-The-Money	0.000143	0.011973	0.009208	0.964370
Longer Maturity	0.000112	0.010565	0.007319	0.989863

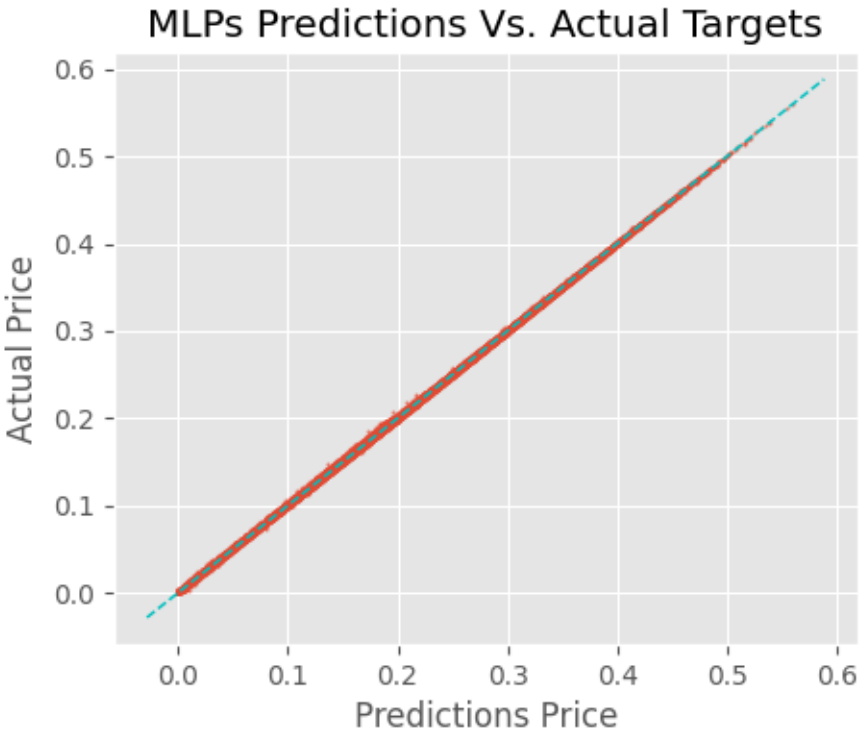


FIGURE 5.4: MLP Performance on in-sample dataset on American bivariate contingent claim

The MLP has shown good performance in terms of our performance measures on all the derivatives considered. Next chapter will compare this MLP model (henceforth referred to as MLP II) with the closed form solutions, binomial model, LSM and LSM MLP (henceforth referred to as MLP I). All the models considered in this section had good performances on the in-sample dataset except for the too simple models; linear regression and 2. degree polynomial regression. The MLP II was superior for out-of-sample data sets compared to the polynomial regression in terms of our performance measures.

The MLP II is not based on a model, it is only trained on the available data. This feature makes the MLP II pricing method versatile, because the MLP could also be used for actual market data or different models to learn patterns. The fact that the method can be extended to real market data is already shown in (Gaspar, Lopes, and Sequeira, 2020). This section showed how the MLP can be trained to pricing derivatives with Black-Scholes theory and that the MLP is preferred over polynomial regression.

Chapter 6

Numerical Investigation and Discussion

This chapter will empirically compare the numerical and analytical methods presented for different types of contingent claims[1]. The underlying model will be the Black-Scholes model, because it has closed form solutions for some European options and is thoroughly researched.

We look at the closed form solutions to the European options compared with the binomial lattice model for pricing. The closed form solution provides a measure of how the binomial lattice model approximates European options in the Black-Scholes model. The binomial model is readily extended to American options, hence, the section also gives an indication of how the binomial model approximates the American option. The American put option is then investigated, where the different numerical methods considered are compared. The last type of option we look at is the American put minimum on two stocks option. After the numerical investigation the pricing methods and model assumptions are discussed.

6.1 European Options

European options are simple in the sense, that they can only be exercised at maturity. Throughout the previous chapters, especially chapter 2 and 3, we have seen closed form solutions for the simple European and exotic European options. The binomial lattice models presented are models that can approximate European options and American options. The closed form solutions and the binomial lattice approach can be compared for European options, where a small deviation between methods for the European options indicates that the lattice approach gives reasonable prices for the American options.

The simplest case is the European call option, where we look at how the CRR model and the MLP II pricing model approximate the B-S call formula. Table 6.1 empirically shows that the CRR model price prediction converges toward the price from the Black-Scholes call formula, which is in line with the theoretical result for the CRR model that it converges to the Black-Scholes model. The MLP II pricing method underprices the European call option, but it performs better for this example than the CRR with 10 and 30 time-steps. Keep in mind we trained the MLP II with the CRR 100 equidistant time-steps model, therefore, we might have seen a better fit with a data set generated with 10^4 time-steps.

[1] The interested reader can see the implementation details in appendix D

TABLE 6.1: CRR, MLP II, and B-S Call Formula Comparison

Comparison of accuracy for the European call option, where the inputs are K=40, $S(0) = 40$, $\sigma = 0.2$, T=1, and r=0.06.

Method	**No. Steps**	**Price**
CRR	10	4.316
	30	4.369
	50	4.380
	100	4.388
	200	4.392
	500	4.394
	1000	4.395
	10000	4.396
MLP II		4.370
Analytic form		4.396

The natural extension of the CRR model is the BEG model, where it is possible to price more exotic options. Section 3.4 showed some closed form solutions for exotic European options. Hence, we have a benchmark for the BEG in these special cases. We choose to look at the computation time for European put minimum with two underlying assets, which has a closed form solution. Closed form solutions make it easy to look at the trade-off between accuracy and computational cost for the BEG method.

TABLE 6.2: BEG Accuracy and Speed

Comparison of speed and accuracy for a European put min option, where the inputs are K=40, $S_1(0) = S_2(0) = 40$, $\sigma_1 = 0.2, \sigma_2 = 0.3$, T=1, $\rho = 0.5$, and r=0.06. Note ms is shorthand for millisecond

Method	**No. Steps**	**Price**	**Time: min:sec.ms**
BEG	10	4.248	0:00.003
	50	4.341	0:00:097
	100	4.352	0:00.591
	200	4.358	0:04.121
	500	4.361	0:59.337
	1000	4.362	9:34.164
Analytic form		4.363	

Tabel 6.2 shows that the algorithm accuracy increases with the number of equidistant time-steps, but the computational speed dramatically slows down when the time-steps increases. Therefore, for exotic options the computational costs become a factor to consider[2]. The BEG method accuracy is also tested on the European call minimum and European call maximum for 100 time-steps, where the BEG method

[2]Note the implementation is written in Python, hence, the code can be improved in terms of computational efficiency. The computations are performed on my laptop with 8GB ram and 8th Generation Intel® Core™ i5 processor

is within 0.13 of the analytic solution (table 6.3).

TABLE 6.3: BEG and Exotic European Options

Valuation of bivariate contingent claims with K=40, $S_1(0) = S_2(0) = 40$, $\sigma_1 = 0.2, \sigma_2 = 0.3$, T=1, $\rho = 0.5$, and r=0.06.

Derivative type	**Method**	**No. Steps**	**Price**
European Call Minimum	BEG	100	2.475
	Analytic form		2.483
European Call Maximum	BEG	100	7.787
	Analytic form		7.800

The above tables show that the binomial pricing model can be used for both univariate and bivariate European contingent claims. The binomial model accuracy is high for European options, hence, we expect a similar good approximation for the American option. The CRR for univariate and BEG for bivariate contingent claims will be used as benchmarks for the American options, where we investigate LSM, MLP I and MLP II pricing methods.

Note that the BEG is not practical for pricing multivariate contingent claims with many underlying assets, because the possible states for the stochastic processes increase exponentially. Therefore, the bivariate contingent claim is considered instead of higher dimensional basket options in section 6.3.

6.2 American Put Option

The American put option has no analytical solution in the Black-Scholes model, thus, numerical methods are required. We present and compare the results for the LSM, MLP I and MLP II pricing methods compared to the CRR model.

The LSM and MLP I pricing method are almost identical, except the MLP tries to utilize deep learning to regress the expected continuation value. Both pricing methods converge to the optimal value process[3], hence, we numerically expect that by increasing the computational burden we approach the true price. To compare the two methods we simulate 10^5 paths for the stock under the assumption that the future price of the stock is lognormal. The LSM and MLP I pricing methods are used on the same simulated paths, but produce a different result each time due to the Monte Carlo simulations. The CRR and MLP II[4] are not random in the sense that the output is deterministic, because both approaches do not involve Monte Carlo simulation. For the LSM and MLP I we assume 50 equidistant exercise dates for each year, where for the CRR we use 1000 equidistant time-step for the stock.

The MLP I require us to set some hyperparameters, where we choose the learning rate $\eta = 0.001$, batch size of 512 and the Adam optimization algorithm. The architecture of the MLP is three layers, where the hidden layers are with 40 neurons and the output layer has one neuron. The activation functions are set to Leaky ReLU

[3] See appendix C.2

[4] Note that we talk about the model after training

with 0.3 negative slope and the trained model is reused at each decision point by inspiration from (Lapeyre and Lelong, 2019). The choices are partly inspired by the work by (Lapeyre and Lelong, 2019) and empirical testing. The regression in the LSM is done with a polynomial regression of degree 10. Remember the MLP II is trained with the same hyperparameters as for the European call option and the CRR with 100 time-steps is used to generate labels.

TABLE 6.4: Valuation of American Put Option

We choose the fixed parameters K=40 and r=0.06

Spot	σ	**T**	**CRR**	**LSM**	**MLP I**	**MLP II**
36	0.2	1	4.487	4.481	4.364	4.584
36	0.2	2	4.848	4.846	4.747	4.649
36	0.4	1	7.109	7.118	6.919	7.090
36	0.4	2	8.508	8.514	8.215	8.487
38	0.2	1	3.257	3.258	3.217	3.094
38	0.2	2	3.751	3.748	3.681	3.638
38	0.4	1	6.154	6.157	6.075	6.172
38	0.4	2	7.675	7.695	7.359	7.605
40	0.2	1	2.319	2.317	2.292	2.114
40	0.2	2	2.900	2.896	2.823	2.779
40	0.4	1	5.318	5.329	5.180	5.274
40	0.4	2	6.923	6.934	6.750	6.839
42	0.2	1	1.621	1.623	1.599	1.494
42	0.2	2	2.217	2.224	2.183	2.167
42	0.4	1	4.588	4.600	4.538	4.548
42	0.4	2	6.250	6.269	6.111	6.197
44	0.2	1	1.113	1.119	1.094	1.000
44	0.2	2	1.694	1.700	1.653	1.678
44	0.4	1	3.954	3.959	3.931	3.949
44	0.4	2	5.647	5.669	5.524	5.649

Table 6.4 shows that the MLP I always predicts a lower price than the LSM, therefore, for our numerical study, the LSM seems to be better than the MLP I in terms of approximating the *"true"* price. The reference is the CRR model, which is a deterministic method. The MLP II trained with the CRR model shows high deviation from the CRR predicted price compared to LSM. The total deviation in absolute distance for the MLP II is 1.56, where LSM deviation is 0.157 for the above table containing 20 prices with different unique input parameters combinations. The MLP II is, however, better in total absolute deviation compared to the MLP I, which has a total absolute deviation of 2.078. This indicates that the MLP II, at this stage of development, is preferred over the MLP I in terms of speed and accuracy. The MLP I and LSM have uncertainty from the Monte Carlo simulation and with 10^5 paths the standard error of the means are 0.0019 and 0.0214. The standard error[5] of the means are calculated by 100 samples[6] for the input parameters T=1, $\sigma = 0.4$, $r = 0.06$,

[5]Sometimes written in short form SE
[6]Denoted with n in the formulas

$S(0) = 36$, and $K = 40$. The empirical distribution mean is calculated by

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

and the standard error of the mean

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}} \quad where\ \sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})}$$

The histogram (figure 6.1) shows the variation in the estimates where the CRR price is the dashed black line. We see that the the MLP I has higher standard error than the LSM and the center of the distribution[7] is lower than the CRR price[8]. The LSM, on the other hand, has less variability and the center[9] is around the CRR price. Hence in term of numerical stability, computational speed and accuracy the LSM is superior for the American put. The reason for the numerical instability for the MLP I is that the optimization algorithm is random on each run compared to the linear model[10].

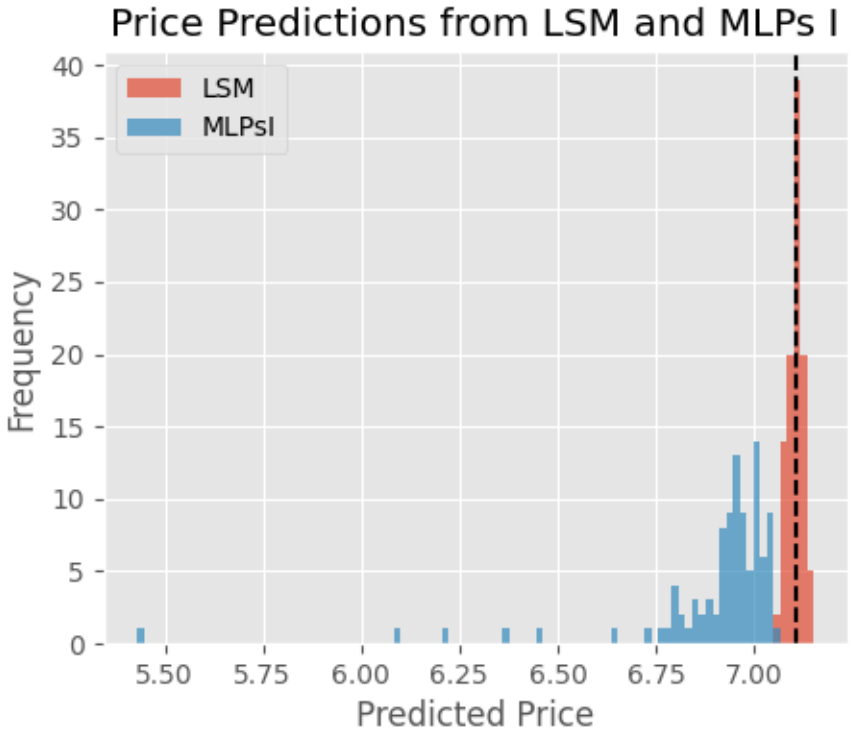


FIGURE 6.1: Predicted prices for American put option with LSM and MLP. Parameters are T=1, $\sigma = 0.4$, $r = 0.06$, $S(0) = 36$, and $K = 40$. Note CRR price is the dashed black line.

[7] Sample mean is 6.9048
[8] CRR price is 7.1094
[9] Sample mean is 7.1074
[10] E.g. polynomial regression where the solution is exact and unique

Improvements of MLP I is needed in order for the method to challenge the existing LSM, both in terms of speed and accuracy for low dimensional problem. Possible improvement of the MLP I could be conducted by a large hyperparameter tuning study. The issue with hyperparameter tuning is that it is computationally expensive. For the MLP I the hyperparameter tuning needs to be searched at every decision point. Another type of network could also be considered, e.g. recurrent neural network. The inferior results from the MLP I to the LSM show that some work still needs to be done. We choose not to consider MLP I and LSM for the next section, because of the poor results from the MLP I pricing method in this section.

6.3 American Put Minimum on two Assets Option

There are many exotic American options to consider, but we choose to focus on the American put minimum on two stocks option, because MLP II is trained to predict prices on this specific option. The other advantage is that we have the deterministic method BEG for the bivariate contingent claim, hence, the MLP II can easily be compared, both for accuracy and speed.

First, we look at the computational cost for the BEG method by increasing the number of time-steps. By looking at table 6.5 it is clear that the computational time is dependent on the number of time-steps. We know from the discussion in section 6.1 that the accuracy increases by increasing the number of time-steps. Weighting the computational cost and accuracy we choose to use the BEG method as reference price with 500 time-steps. Note, that when we generated the labels for the MLP II pricing method we used 50 equidistant time-steps. The reason for using 50 time-steps is that we simulated 300.000 data points, which is a computational heavy task. A MLP II model trained with 500 time-steps would probably predict the price from the BEG method with 500 steps better, but it was not computationally feasible. This creates an expectation that the option priced with MLP II will have a bias price and therefore, we will also include the BEG with 50 time-steps in the comparison.

TABLE 6.5: BEG for the American Bivariate Contingent Claim

Comparision of speed for the American put minimum on two assets option, where the inputs are K=40, $S_1(0) = S_2(0) = 40$, $\sigma_1 = 0.2, \sigma_2 = 0.3$, T=1, $\rho = 0.5$ and r=0.06.
Note ms is shorthand for millisecond

Method	**No. Steps**	**Price**	**Time: min:sec.ms**
BEG	10	4.524	00:00.006
	50	4.594	00:00.250
	100	4.602	00:01.837
	200	4.605	00:14.025
	500	4.608	03:35.039
	1000	4.609	28:32.584
MLP II		4.426	00:00.0003

Table 6.5 shows that the price for the American put minimum is greater than the European put minimum from table 6.2. Additionally, it is clear that the computational time increases for the American option, because we have to compare intrinsic

value and expected continuation value at each node. Both results are good sanity checks.

For visualization we plot the BEG50, BEG500, and MLP II for varying spots with fixed K=40, $\sigma_1 = 0.2$, $\sigma_2 = 0.3$, T=1, $\rho = 0.5$, and r=0.06. The figure shows that the BEG500[11] and BEG50 are close to each other. The total absolute deviation is 0.142 for the 21 prices, which shows that the BEG50 is suffient for training our MLP. The total absolute deviation for the MLP II with the BEG500 is 3.716 and with the BEG50 3.713. This is depicted in figure 6.2, where the MLP underprices the option for the in-sample domain with varying spot. To compare the computational speed the MLP II took around 00:00.000349 to generate one price, which is considerable faster than the BEG model with 10 time-steps. Hence, we see that we loose some accuracy with the MLP II, but the pricing method is fast.

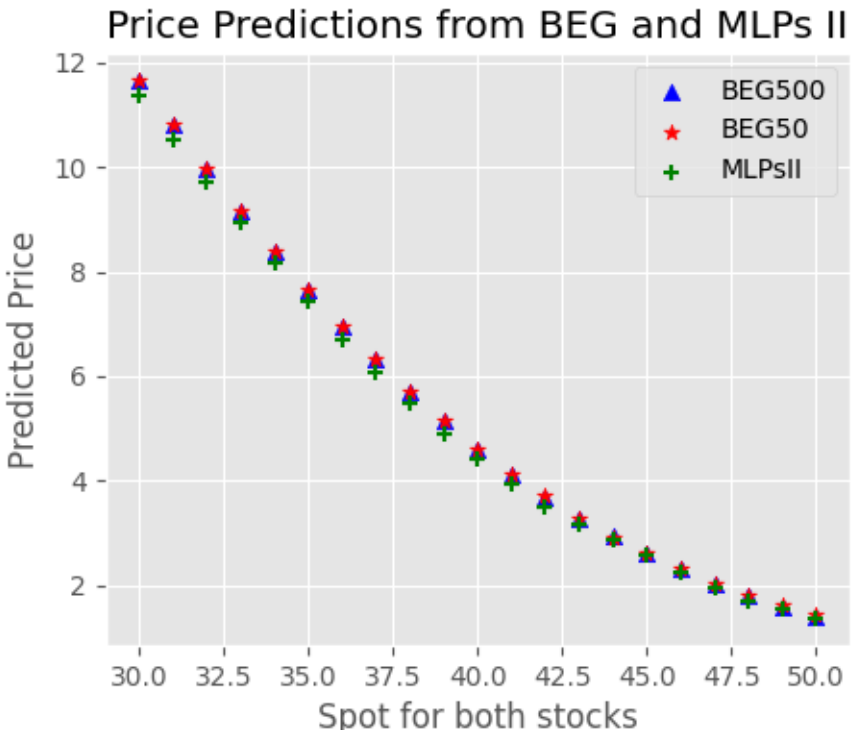


FIGURE 6.2: Scatter plot of predicted prices for American put minimum on two assets option with BEG and MLP II. Parameters are T=1, $\sigma_1 = 0.2$, $\sigma_2 = 0.3$, $r = 0.06$, $K = 40$ and $\rho = 0.5$. Note BEG is showed for 50 time-steps and 500 time-steps.

The MLP might provide a better estimate by simulating more data samples and make the model more complex, but our hyperparameter tuning study was not conclusive on this point. Hyperparameter tuning could also be further investigated, e.g. looking at depth and width of the network. The real power of pricing with MLP II is when the computational burden becomes an issue for fast pricing.

[11] BEG method with 500 time-steps

6.4 Discussion of Pricing Methods

The idea behind using neural networks instead of the linear model, is that the neural networks scale better for high dimensional problems. We have only considered univariate and bivariate contingent claims, because in these cases we have a deterministic alternative in the binomial lattice models. This gives a good indication of how the model predicts. Besides, if the model does not perform well for low dimensional problems, we would not expect the model to be any better for more complex multivariate contingent claims.

Remember, the binomial lattice model also has its limitations in terms of computational resources, because the computational burden scales exponentially with the number of underlying risky assets[12]. The LSM is versatile and can be used for most derivatives, because it relies on Monte Carlo simulation and regression. For considering multivariate contingent claims the LSM is readily useful, but the linear model for regression suffers the curse of dimensionality for basket options with many underlying risky assets. We have in mind that a neural network does not suffer from the curse of dimensionality, hence, the method, compared to the linear model, has advantages when considering multivariate contingent claims. The idea of neural network should work in theory[13], but the numerical results shown in this chapter are not satisfying for the MLP I in the univariate case.

The MLP II is somewhat better, but it still relies on existing option pricing methods or real market data. Therefore, it does not solve the curse of dimensionality for basket options with many underlying assets. The MLP II however, has the advantage of the increased speed after training, which could be beneficial in some circumstances. The MLP could also be used on real data and it is versatile enough to capture the volatility shew[14] for equity options. One drawback is that there should be enough data samples, which is relevant if the method is used on real market data. Another drawback is that the trained model would need to be calibrated regularly, because financial scenarios can change dramatically, e.g. the financial crisis in 2007-2009.

In the training phase of the neural networks the derivatives of the model parameters are used for minimizing the loss function. The derivatives are calculated efficiently with backpropagation, which is important for risk management. Potentially, the neural networks can speed up the risk management of the derivative books and give real time risks. Deep learning for risk management is investigated in (Savine, 2019). It should be mentioned, that the above-mentioned methods cannot only be used for equity options, the LSM is e.g. widely used for fixed income[15].

The results from the MLP I were a bit discouraging, because it was inferior to the LSM. The computational cost is higher for the MLP I for low dimensional problems, hence, an accuracy on the same level as for the LSM would make it attractive to investigate for higher dimensional problems. One explanation could be wrong hyperparameters for the model, which would require a time consuming hyperparameter search. The grid search might have its shortcomings in this aspect, because

[12] The same can be said about the PDE methods
[13] Universal approximation theorem for MLP
[14] Explanation below in section "Discussion of the Black-Scholes model"
[15] E.g. Libor Market Model

it requires some knowledge about possible well-suited hyperparameters for the optimal stopping problem. A random search or Bayesian search could be beneficial in this aspect, but it would also require many computational resources.

The MLP II lack some precision compared to LSM for the univariate contingent claim. For the bivariate the MLP II again showed a loss of accuracy, but showed additional speed. A larger hyperparameter study could also be conducted here, a different method to generate labels or a bigger data set could have been sampled. E.g. the article (Ferguson and Green, 2018) shows good result for an European basket option with 6 underlying assets by simulating a data set of 500M samples.

6.5 Discussion of the Black-Scholes model

For pricing we utilize the Black-Scholes theory, where we assume constant volatility, the interest rate is constant through time, and future stock prices are lognormal distributed. It is well-known that the volatility is not constant, in fact, it depends on the maturity and strike. The dependency on strike and maturity can be modeled with a volatility surface. Looking only at one dependency the implied volatility for equity options is a decreasing function of the strike over spot $\frac{K}{S(0)}$ for real data (p. 458 (Hull, 2017)). This phenomenon is known as volatility shew for equity options.

A possible reason for the volatility shew is that investors are more concerned with falling stock prices than rising prices, hence, the volatility is instantaneously negatively correlated with the stock price. To overcome the issue about assuming constant volatility, a model with stochastic variance can be considered. The model becomes more complex with an extra stochastic variable, where for simplicity we mention the two factor Heston model. The basic model is given by the stock follow the SDE:

$$dS(t) = \alpha S(t)dt + \sqrt{V(t)}S(t)dW_S(t)$$

And the stochastic variance process $V(t)$ is the solution to the SDE:

$$dV(t) = a(\theta - V(t))dt + \epsilon\sqrt{V(t)}dW_V(t) \quad \textit{where } a > 0, \theta > 0, \epsilon > 0, \textit{ and } V(0) > 0$$

Where $W_S(t)$ and $W_V(t)$ have correlation ρ. The interpretation of the constants is:

- θ is long run average price variance
- a is the rate which $V(t)$ reverts to θ
- ϵ is the volatility of the volatility

The implication of ϵ is that $V(t)$ is more volatile when volatility is high. The correlation between the stock and variance process is often modeled with a negative correlation. The negative correlation between the two processes displays the real market phenomenon of volatility shew for equity options, because when the stock price drops, volatility increases. Assuming stochastic volatility the Heston model overcomes the issue with volatility shew. Another model trying to solve the non-constant volatility is e.g. "Constant Elasticity of Variance model", but there are numerous others[16].

[16] E.g "Merton's Mixed Jump-Diffusion Model" and "Variance-Gamma Model"

The constant risk-free interest rate through time assumption can also be discussed, because the interest rate is not constant for real market behavior. We did not investigate the American call, because it coincides with the European call for positive interest rate. Today's market interest rates are negative for the Eurozone, which was probably unheard before the last decade's financial events. Therefore, the decision to assume positive constant interest rate through time is set for convenience.

The Black-Scholes model assumes that the underlying risky assets evolves as a GBM, where the distribution of possible stock prices at the end of any interval is lognormal. The model is convenient, because the GBM has a analytical solution[17]. If we wanted to investigate arithmetic mean basket option the Bachelier model would be more convenient, because the future stock price is normal distributed.

$$dS_i = \sigma_i dW_i$$

This assumption would simplify the pricing problem of arithmetic basket options, because the sum of normals provides a multivariate normal distribution. The basket option problem in the Bachelier is then essentially one dimensional. This is similar to the the geometric mean basket option for the Black-Scholes model (section 3.4.1). A disadvantage with the Bachelier model is that it can lead to negative stock values, which is not realistic.

The Black-Scholes model has some drawbacks where you can question the assumptions, but the model is convenient to use for comparison of pricing methods. By above discussion, we stress that we do not necessarily believe that the Black-Scholes model is the true model for real market behavior. The purpose is rather to investigate pricing methods in a convenient model.

[17]SDE does not always have a analytical solution

Chapter 7

Conclusion and Further Investigation

7.1 Conclusion

We have seen different methods for pricing equity options within the Black-Scholes world. We have specifically focused on the European call, exotic European, American put and American put minimum on two stocks options. The closed form solutions, the classical binomial lattice model and LSM were provided for investigating the potential of using deep learning in option pricing theory.

The numerical investigation showed that MLP I and MLP II pricing methods were inferior in accuracy for univariate and bivariate contingent claims, but the MLP II had an enhanced speed compared to the classical methods after training. The results for the MLP I were somewhat discouraging, but the theory tells us that we can improve our training algorithm to perform on the same level as LSM. The CRR model was extended to the BEG method with two underlying stocks, which we saw provided a good approximation for European bivariate contingent claims. The BEG method has its limitations in terms of pricing a basket option with many underlying stocks, which can be solved with LSM or MLP I.

To sum up, the MLP I needs hyperparameter tuning, but the underlying theory and idea gives hope for finding a better model for high dimensional data than the classical LSM. The BEG has its limitation in high dimensions. Hence, LSM is preferred for higher dimensions. The MLP II showed better performance than for polynomial regression at supervised learning. Furthermore, the MLP II can be beneficial in terms of computational speed, but has lower precision for univariate contingent claims than the LSM, compared to our CRR benchmark price.

The classical methods approximate the true price in the Black-Scholes model well. The deep learning methods lack the accuracy of the classical methods in the Black-Scholes model, hence, there is still some work to be done for pricing derivatives with deep learning.

7.2 Further Investigation

The code library is written in Python, which could be optimized in terms of computational speed with e.g. Julia or C++. The advantage of Python is the libraries for machine learning and especially deep learning. The code could also be more generalized, not only to handle specific option contracts by using the object oriented

programming paradigm instead of procedural programming approach. The code was run on my CPU, but the speed could also be improved by using the GPU.

Hyperparameter tuning for the MLP would be interesting to investigate further with e.g. the more advanced method; Bayesian hyperparameter tuning. The article from (Liaw et al., 2018) looks at how hyperparameter tuning can be easy implemented, which is worth looking at. The data sets could have been bigger or the labels could have been generated with a different method. Another interesting aspect to investigate is calculating the derivatives of the pricing function, which produces the risks for the derivative books. It looks like MLP models are superior for calculating risks in terms of speed to the classical methods, hence, there is potentially an additional gain when using deep learning.

Appendix A

Stochastic Calculus and Probability Theory

Theorem A.1. ***Itô's formula multidimensional:*** *Let the n-dimensional process X have dynamics given by:*

$$dX(t) = \mu(t)dt + \sigma(t)dW(t) \tag{A.1}$$

Then the process $f(t, X(t))$ has stochastic differential given by:

$$df(t,X(t)) = \frac{\partial f(t,X(t))}{\partial t}dt + \sum_{i=1}^{n}\frac{\partial f(t,X(t))}{\partial x_i}dX_i(t) + \frac{1}{2}\sum_{i,j=1}^{n}\frac{\partial^2 f(t,X(t))}{\partial x_i \partial x_j}dX_i(t)dX_j(t) \tag{A.2}$$

Note:

$$dW_i(t)\cdot dW_j(t) = \begin{cases} \rho_{ij}dt & \text{For correlated Weiner processes} \\ 0 & \text{For independent Weiner processes} \end{cases}$$

(p. 58-60 (Björk, 2009))

Theorem A.2. ***The Martingale Representation Theorem:*** *Let W be a k-dimensional Wiener process, and assume that the filtration is defined by:*

$$\mathcal{F}_t = \mathcal{F}_t^W \quad t \in [0,T]$$

Let M be any $\mathcal{F}_t$-adapted martingale. Then there exist uniquely determined $\mathcal{F}_t$-adapted processes $h_1, \ldots, h_k$ such that M has the representation

$$M(t) = M(0) + \sum_{i=1}^{k}\int_0^t h_i(s)dW_i(s) \quad t \in [0,T]$$

if the martingale M is square integrable, then $h_1, \ldots, h_k$ are in L^2

(p. 161 (Björk, 2009)).

Theorem A.3. ***The Girsanov Theorem:*** *Assume the probability space $(\Omega, \mathcal{F}, P, (\mathcal{F}_t^{\bar{W}})_{t\in[0,T]})$ and let the Girsanov kernel ϕ be any d-dimensional adapted column vector process. Choose a fixed T and define the likelihood process L on $[0,T]$ by:*
$dL(t) = \phi(t)^T L(t)d\bar{W}(t)$
$L(0) = 1$.
Assume that $E^P[L(T)] = 1$ and define the new probability measure Q on $\mathcal{F}_T$ by:

$$L(T) = \frac{dQ}{dP} \quad \text{on } \mathcal{F}_T$$

Then

$$d\bar{W}(t) = \phi(t)dt + dW(t) \tag{A.3}$$

Where $W(t)$ is the Q-Wiener process and $\bar{W}(t)$ is the P-Wiener process.

(p. 164 (Björk, 2009))

Theorem A.4. ***The Converse of the Girsanov Theorem:*** *Let $\bar{W}$ be k-dimensional standard P-Wiener process (i.e. zero drift and unit variance independent components) on $(\Omega, \mathcal{F}, P, (\mathcal{F}_t^{\bar{W}})_{t\in[0,T]})$. Assume that there exits a probability measure Q such that $Q << P$ on $\mathcal{F}_T^{\bar{W}}$. Then there exists an adapted process ϕ such that the likelihood process L has dynamics*

$$\begin{aligned} dL(t) &= \phi^T(t)L(t)d\bar{W}(t) \\ L(0) &= 1 \end{aligned}$$

(p. 168 (Björk, 2009))

Definition A.5. Stopping time: A random variable $\tau : \Omega \to [0, \infty]$ is called a Markov time if

$$(\tau \leq t) \in \mathcal{F}_t \quad \forall t \geq 0$$

A Markov time is called a stopping time if $\tau < \infty$ P-a.s.

(p. 27 (Shiryaev and Peskir, 2006))

Definition A.6. Snell envelope: Consider a fixed process Y.

- We say that a process X dominates the process Y if $X_n \geq Y_n$ $P - a.s.$ $\forall n$.
- Assuming that $E[Y_n] < \infty$ $\forall n \leq T$, the Snell envelope S, of the process Y is defined as the smallest supermartingale dominating Y. More precisely: S is a supermartingale dominating Y, and if D is another supermartingale dominating Y, then $S_n \leq D_n$ $P - a.s.$ $\forall n$.

(p. 380 (Björk, 2019))

Theorem A.7. ***The Snell Envelope Theorem:*** *The optimal value process V is the Snell envelope of the gain process G.*

(p. 381 (Björk, 2019))

Appendix B

Code Implementation

The code is available at my GitHub repository https://github.com/MrPPL/DeepPricing. The code is written in the Python directory and the following files are used:

- European call and put option:
  ClosedForm directory and the file closedEuro.py
- Exotic European options:
  ClosedForm directory and the file rainbow2Dim.py
- CRR:
  Directory BinomialModel and file BinoHull.py
- BEG:
  Directory BinomialModel and file BEGTwoDim.py
- LSM:
  Directory DeepStopping and file AmericanPut.py[1]
- MLP I:
  Directory DeepStopping and file AmericanPut.py [2]
- MLP II:
  Directory DeepLearning/hirsa19/evaluation and file MLPsIIPricing.py

[1]Note that the LSM is implemented by using luphord GitHub repository https github.com/luphord/longstaff_schwartz

[2]Note that the MLP is my own build upon the LSM method

Appendix C

Additional Material for CRR, LSM, and MLP I

C.1 Moment Matching CRR

In the CRR the stock multiplication factor for up and down movement is chosen to match the two first moments of the lognormal distribution. By matching the first moments the underlying discrete binomial stochastic process for the stock converge toward the continuous time lognormal distribution for sufficient large equidistant time-steps N. Hence, the CRR model will coincide with the Black-Scholes model.

The SDE for the Black Scholes under the equivalent martingale measure Q:

$$dS(t) = rS(t)dt + \sigma S(t)dW(t)$$

Using Itô's lemma:

$$d\ln(S(t))) = (r - \frac{1}{2}))dt + \sigma dW_t \tag{C.1}$$

The solution of equation C.1 is then:

$$S(t) = S(0)\exp\left((r - \frac{1}{2}\sigma)t + \sigma W(t)\right)$$

Note that $W(t) \sim \mathcal{N}(0, t)$ implies:

$$\ln(\frac{S(t)}{S(0)}) \sim \mathcal{N}((r - \frac{1}{2}\sigma^2)t, \sigma^2 t)$$

The two first moments for the lognormal distribution are then:

$$\begin{aligned} E[\frac{S(t)}{S(0)}] &= \exp(rt) \\ E[(\frac{S(t)}{S(0)})^2] &= \exp(t \cdot (2r + \sigma^2)) \end{aligned} \tag{C.2}$$

The above derivation can be used for any time interval.

The binomial lattice model has two discrete outcomes from each state and the moments are:

$$\begin{aligned} E[\frac{S(t_{n+1})}{S(t_n)}] = u \cdot Q(\frac{S(t_{n+1})}{S(t_n)} = u) + d \cdot Q(\frac{S(t_{n+1})}{S(t_n)} = d) = u \cdot q + d \cdot (1-q) \\ E[(\frac{S(t_{n+1})}{S(t_n)})^2] = u^2 \cdot q + d^2 \cdot (1-q) \end{aligned} \tag{C.3}$$

We match the moments (equations (C.2) and (C.3)) and get two equations with two unknowns, where the time-step is chosen to be Δt

$$\begin{aligned} \exp(r\Delta t) &= u \cdot q + d \cdot (1-q) \quad (i) \\ \exp(\Delta t \cdot (2r + \sigma^2)) &= u^2 \cdot q + d^2 \cdot (1-q) \quad (ii) \end{aligned}$$

Multipling (i) with u+d and recognizing (ii):

$$\begin{aligned} (u+d)\exp(r\Delta t) &= u^2 \cdot q + d^2 \cdot (1-q) + ud \\ \Rightarrow (u+d)\exp(r\Delta t) &\overset{(ii)}{=} \exp(\Delta t \cdot (2r + \sigma^2)) + ud \end{aligned}$$

Remember, we choose $u = \frac{1}{d}$ and arrive at a quadratic equation through algebra.

$$u^2 - u\Big(\exp(-r\Delta t) + \exp(\Delta t(r + \sigma^2))\Big) + 1 = 0$$

We are interested in that the binomial model converge toward the Black-Scholes model. Therefore, we are looking at small time increment Δt. This justify the Taylor approximation of the exponential function around zero.

$$\exp(r\Delta t + \sigma^2 \Delta t) \approx 1 + (r + \sigma^2)\Delta t + O(t^2)$$

By Taylor approximation we arrive at a simpler quadratic equation:

$$u^2 - u(2 + \sigma^2 \Delta t) + 1 = 0$$

Solving the quadratic equation above gives:

$$\begin{aligned} u &= 1 + \frac{1}{2}\sigma^2 \Delta t \pm \frac{\sqrt{(2 + \sigma^2 \Delta t)^2 - 4}}{2} \\ &= 1 + \frac{1}{2}\sigma^2 \Delta t \pm \frac{\sqrt{(4 + \sigma^4 \Delta t^2 + 4\sigma^2 \Delta t - 4}}{2} \\ &= 1 + \frac{1}{2}\sigma^2 \Delta t \pm \frac{1}{2}\sigma\sqrt{\Delta t}\sqrt{\sigma^2 \Delta t + 4} \\ &\approx 1 + \frac{1}{2}\sigma^2 \Delta t \pm \sigma\sqrt{\Delta t} \\ &\approx \exp(\pm\sigma\sqrt{\Delta t}) \end{aligned}$$

Both approximations exploit that the time-step is small.

C.2 Convergence for LSM and MLP I

C.2.1 LSM

In the rigorous approach in (Clément, Lamberton, and Protter, 2001) they show convergence results for the optimal value process or the Snell envelope U. We will present that $U(0)^{m,K}$ converges almost surely to $U(0)^m$ for K goes to infinity, i.e. the approximate value process by simulation and regression on a finite set of functions converge to the approximated value process with truncated orthogonal basis by letting the sample size go to infinity. Furthermore, it can be shown that $U(0)^m$ converges to $U(0)$ for m goes to infinity. The second result is that the regressed value function converges to the expected continuation value by letting the number of basis functions go to infinity. The latter result is shown using the expected continuation values.

Theorem C.1. *Assume the sequence $(e_j(S(t_n)))_{j\geq 1}$ is total in $L^2(\sigma(S(t_n)))$ for $n = 1, \dots, N-1$. Then for $n = 0, \dots, N$ we have*

$$\lim_{m\to+\infty} E^Q[G(S(\tau_{t_n}^{[m]}))|\mathcal{F}_{t_n}] = E^Q[G(S(\tau_{t_n}))|\mathcal{F}_{t_n}]$$

in L^2

Proof. The proof is given by induction, where the orthogonal basis is total in L^2 is important because $||P_{t_n}^m(E^Q[G(S(\tau_{t_{n+1}}))|\mathcal{F}_{t_n}]) - E^Q[G(S(\tau_{t_{n+1}}))|\mathcal{F}_{t_n}]||_2 \to 0 \quad for\ m \to \infty$. (more details on p. 6-7 (Clément, Lamberton, and Protter, 2001)) ■

The former result is also shown in (Clément, Lamberton, and Protter, 2001).

Theorem C.2. *Assume the sequence $(e_j(S(t_n)))_{j\geq 1}$ is total in $L^2(\sigma(S(t_n)))$ for $n = 1, \dots, N-1$ and if $\sum_{j=1}^m \lambda_j e_j(S(t_n)) = 0$ a.s. then $\lambda_j = 0$ for $n = 1, \dots, N-1$, $m \geq 1$ and $j = 1, \dots, m$. Furthermore assume that $Q(\alpha_{t_n} \cdot e(S(t_n)) = G(S(t_n))) = 0$.*
Then $U^{m,K}(0)$ converges almost surely to $U^m(0)$ as K goes to infinity.
The proof is out of scope for this thesis, see the article (Clément, Lamberton, and Protter, 2001) for a proof in details.

The two convergence results show the convergence for the LSM algorithm, hence, the LSM will approximate the optimal value process well for sufficiently large sample sets and enough basis functions.

C.2.2 MLP I

The MLP regression method enjoys the same convergence results presented for the LSM algorithm, i.e.

Theorem C.3. *Assume that*

$$E[\max_{0\leq t_n\leq T} |G(S(t_n))|^2] < \infty$$

. Then $\lim_{p\to\infty} E^Q[G(S(\tau_{t_n}^p))|\mathcal{F}_{t_n}] = E[G(\tau_{t_n})|\mathcal{F}_{t_n}]$ in $L^2(\Omega)$ for all $n \in \{1, 2, \dots, N\}$
Proof p. 7-8 (Lapeyre and Lelong, 2019)

The above theorem states convergence of the neural network approximation, i.e. $U^p(0) \to U(0)$ which is similar to the convergence result for LSM.

Theorem C.4. ***Strong law of large numbers:*** *Assume*

A1: For every $p \in \mathbb{N}$, $p > 1$, *there exist* $q \geq 1$ *and* $\kappa_p > 0$ *such that*

$$\forall s \in \mathbb{R}^R, \forall \theta \in \Theta_p, \quad |\Psi_p(s,\theta)| \leq \kappa_p(1 + |s|^q)$$

Moreover $\forall n \in \{1, 2, \ldots, N-1\}$, *a.s. the random functions* $\theta \in \Theta_p \mapsto \Psi_p(S(t_n), \theta)$ *are continuous. Note* Θ_p *is a compact set, hence, the continuity is uniform.*

A2: For q defined in A1, $E^Q[|S(t_n)|^{2q}] < \infty \quad \forall n \in \mathbb{N} \cup 0$

A3: $\forall p \in \mathbb{N}$, $p > 1$ *and* $\forall n \in \{1, 2, \ldots, N-1\}$,

$$P(S(t_n) = \Psi_p(S(t_n); \theta^p_{t_n}) = 0$$

A4: $\forall p \in \mathbb{N}$, $p > 1$ *and* $\forall n \in \{1, 2, \ldots, N-1\}$, *if* θ^1 *and* θ^2 *solves*

$$\underset{\theta \in \Theta_p}{\operatorname{argmin}} E^Q[|\Psi_p(S(t_n); \theta) - S(\tau^p_{t_{n+1}})|^2]$$

then $\Psi_p(s, \theta^1) = \Psi_p(s, \theta^2)$ *for almost all x.*

If A1-A4 hold, then for $\zeta \in \{1, 2\}$ *and every* $n \in \{1, 2, \ldots, N\}$

$$\lim_{K \to \infty} \frac{1}{K} \sum_{k=1}^{K} \left(G(S(\hat{\tau}^{k,p,K}_{t_n})) \right)^{\zeta} = E[\left(G(S(\tau^p_{t_n})) \right)^{\zeta}] \quad a.s. \tag{C.4}$$

Proof p. 13-14 (Lapeyre and Lelong, 2019)

The last result is the strong law of large numbers for the value function, i.e. $U^{p,K}(0) \to U^p(0)$ *a.s.* $\quad for\ K \to \infty$

C.3 LSM Lower Bound

The LSM approach gives a lower bound for the true price of the option given optimal stopping choice:

Proposition C.5. ***Lower Bound To True Value:*** *For any finite choice of M, K, and vector* $\theta \in \mathbb{R}^{M \times (K-1)}$ *representing the coefficients for the M basis functions at each of the K-1 early exercise dates, let* $LSM(\omega; M, K)$ *denote the discounted cash flow resulting from the following the LSM rule of exercising when the immediate exercise value is positive and greater than or equal to* $\hat{F}_M(\omega_l; t_k)$ *as defined by* θ. *Then the following inequality almost surely holds,*

$$V(X) \geq \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} LSM(\omega_i; M, K)$$

(p. 124 (Longstaff and Schwartz, 2001))

Appendix D

Additional Tables and Figures

D.1 Tables

TABLE D.1: Polynomial Regression for European Call

Polynomial regression result for the European call option

Data set	**Measure**	**1. degree**	**2. degree**	**3. degree**	4. degree	**5. degree**	6. degree
In-Sample	MSE	0.000636	0.000069	0.000013	0.000004	0.000002	0.000001
	RMSE	0.025212	0.008316	0.003624	0.002052	0.001414	0.000958
	MAE	0.018326	0.006141	0.002561	0.001280	0.000867	0.000591
	R^2	0.936628	0.993105	0.998691	0.999580	0.999801	0.999909
Long Maturity	MSE	0.002662	0.001196	0.001654	0.003956	0.012255	0.043361
	RMSE	0.051593	0.034577	0.040666	0.062894	0.110702	0.208233
	MAE	0.041232	0.026287	0.028371	0.039932	0.064402	0.111190
	R^2	0.818143	0.918316	0.887014	0.729744	0.162742	-1.962442
Out-of-Money	MSE	0.005772	0.000767	0.000839	0.000944	0.001812	0.004423
	RMSE	0.075973	0.027694	0.028960	0.030724	0.042568	0.066506
	MAE	0.060936	0.022203	0.020288	0.020542	0.027125	0.041315
	R^2	-2.377251	0.551246	0.509280	0.447668	-0.060261	-1.588030

TABLE D.2: Grid Search for American Bivariate Contingent Claim

Hyperparameter tuning of dataset size and batch size for American put minimum

	Data set Size	η	**Batch Size**	**Train Loss**
	300K	0.0001	64	1.35215850605164E-06
	300K	0.001	64	2.1698210730392E-06
	300K	0.0001	8	2.73427281172189E-06
	300K	0.0001	256	2.94285678137385E-06
	300K	0.001	256	3.68489895663515E-06
	100K	0.0001	64	4.08099731430411E-06
	100K	0.001	64	4.81692904941156E-06
	100K	0.0001	256	5.18195474796812E-06
	300K	0.0001	1024	5.53397603653139E-06
	300K	0.001	8	5.79783136345213E-06
	300K	0.001	512	5.94204675508081E-06
	300K	0.0001	512	6.09576545684831E-06
	100K	0.0001	8	6.11733685218496E-06
	100K	0.001	256	7.29860903447843E-06
	100K	0.001	8	7.64047854318051E-06
	300K	0.001	1024	8.88059821591014E-06
	100K	0.0001	512	1.00699153335881E-05
	100K	0.001	512	1.02811236502021E-05
	300K	0.01	256	1.08374288174673E-05
	100K	0.001	1024	1.32220848172437E-05
	100K	0.0001	1024	1.34270530907088E-05
	300K	0.01	512	1.47791615745518E-05
two assets	100K	0.01	256	1.87811801879434E-05
	100K	0.01	512	1.94587100850185E-05
	300K	0.01	64	2.06707663892303E-05
	100K	0.01	64	2.37144977290882E-05
	300K	0.01	1024	2.76203390967567E-05
	100K	0.01	1024	3.23796266457066E-05
	1K	0.0001	8	7.90390986367129E-05
	1K	0.001	8	0.000115903450933
	1K	0.001	64	0.0001487621048
	1K	0.0001	64	0.000176786270458
	1K	0.01	64	0.000234621096752
	1K	0.01	8	0.000340490310919
	1K	0.001	512	0.000384355953429
	1K	0.01	512	0.000385054125218
	1K	0.01	256	0.000567226845305
	1K	0.0001	256	0.000780252506956
	1K	0.001	256	0.000782921211794
	1K	0.0001	512	0.002630036789924
	1K	0.0001	1024	0.006931486539543
	1K	0.001	1024	0.012805146165192
	1K	0.01	1024	0.053760576993227
	100K	0.01	8	0.11506936699152
	300K	0.01	8	35609.6484375

D.2 Figures

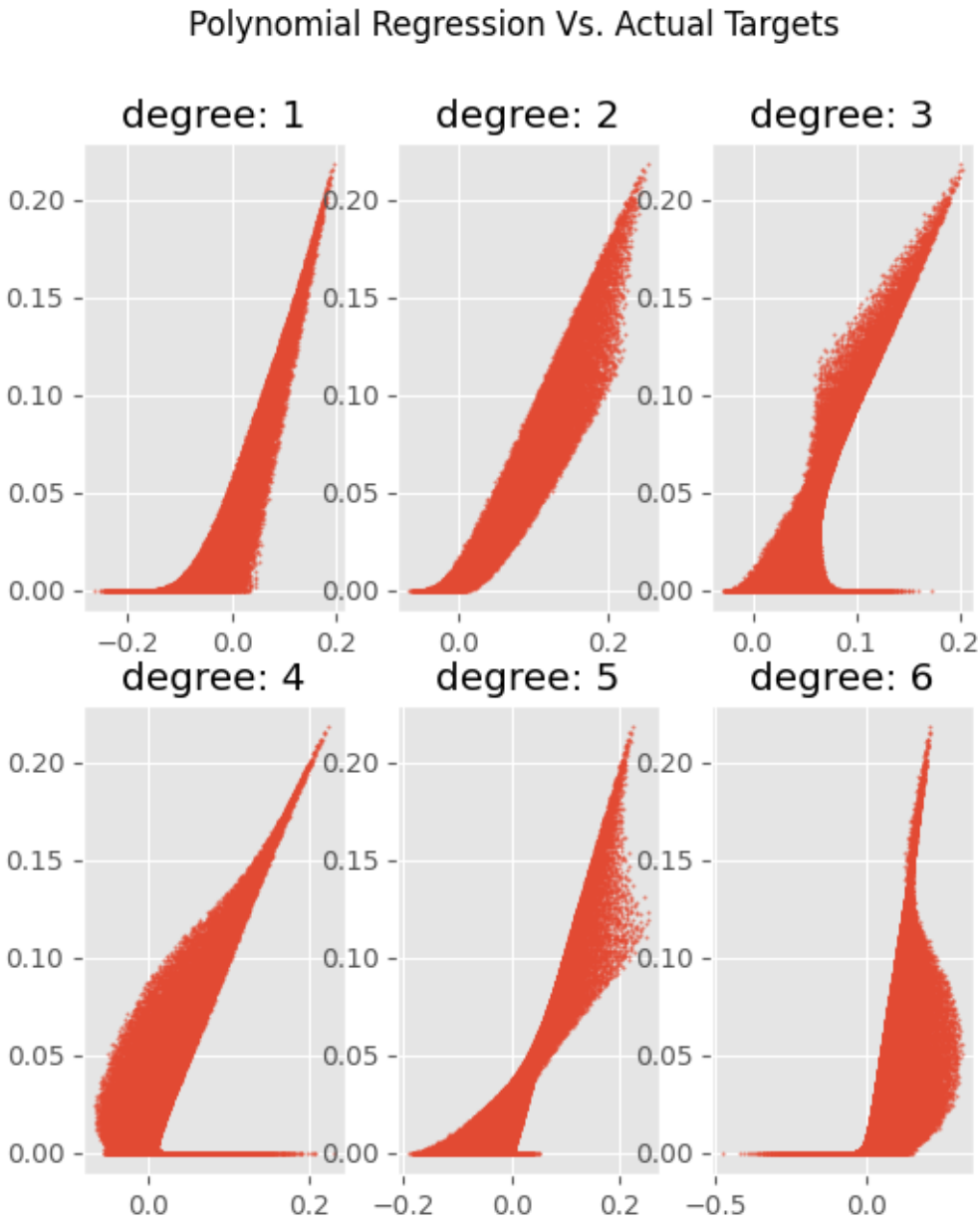


FIGURE D.1: Polynomial regression performance for out of money data set on the European call

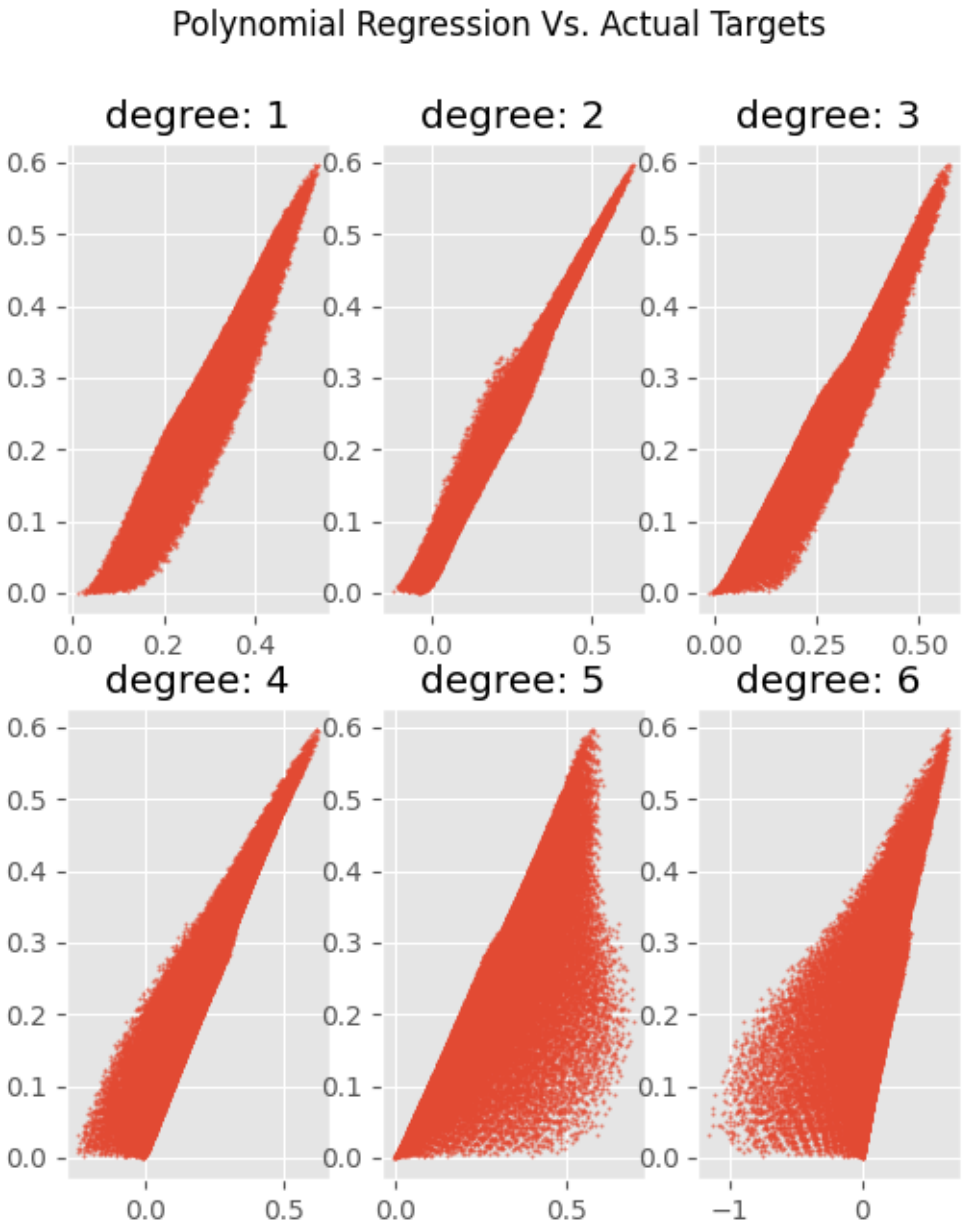


FIGURE D.2: Polynomial regression performance for long maturity data set on European call

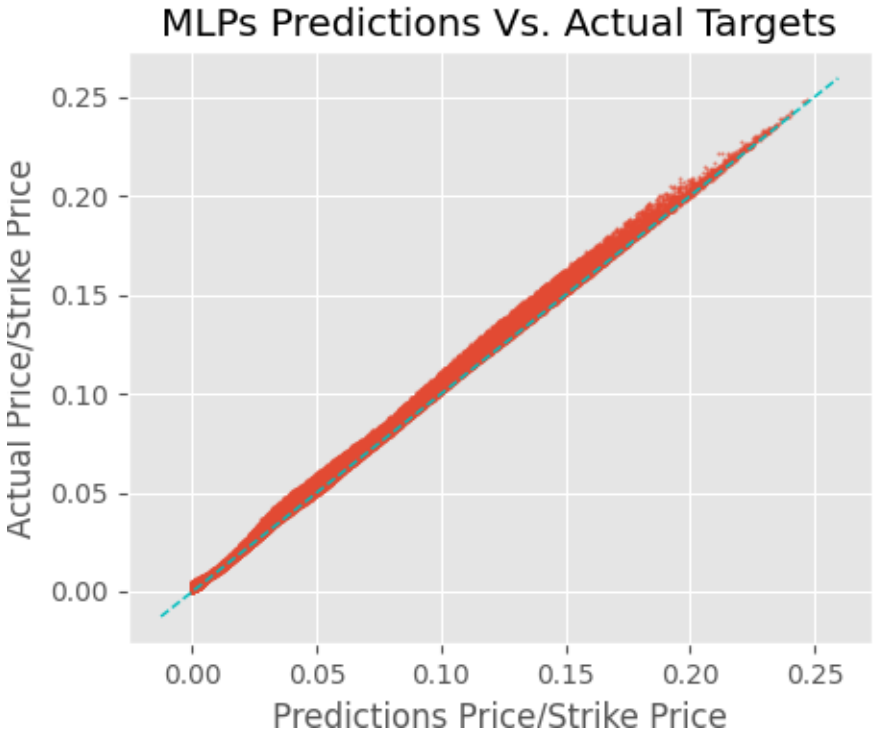


FIGURE D.3: MLP performance for in-the-money data set on American put

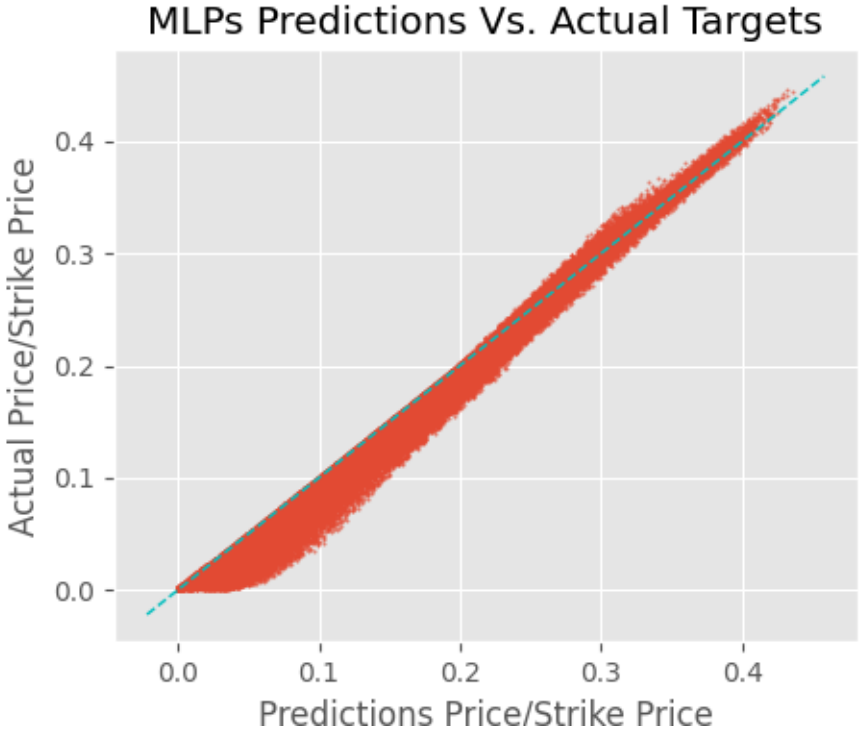


FIGURE D.4: MLP performance for long maturity data set on American put

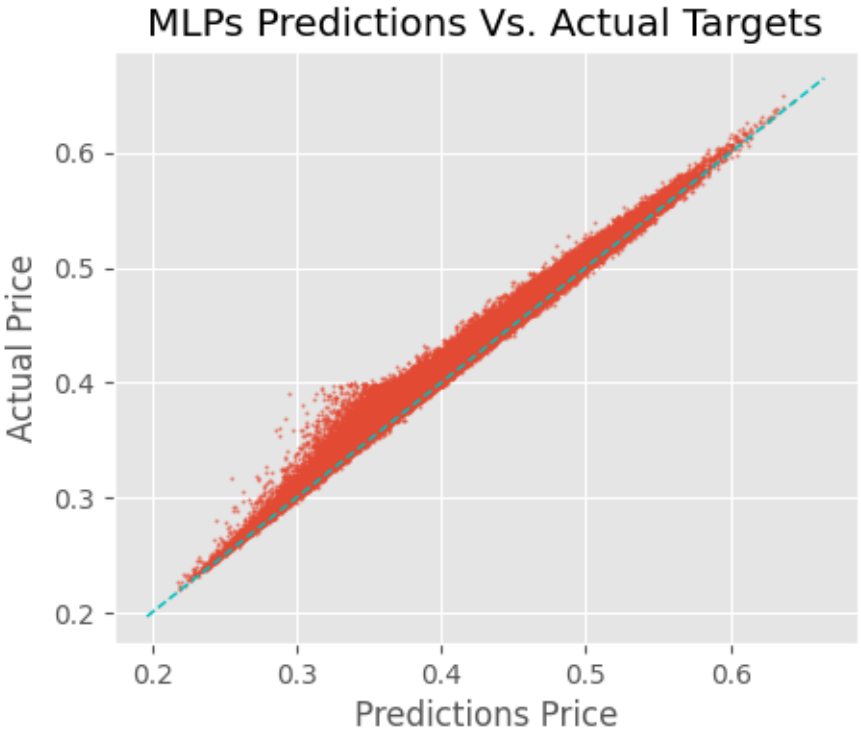


FIGURE D.5: MLP performance on in-the-money data set on American put on minimum of two stocks

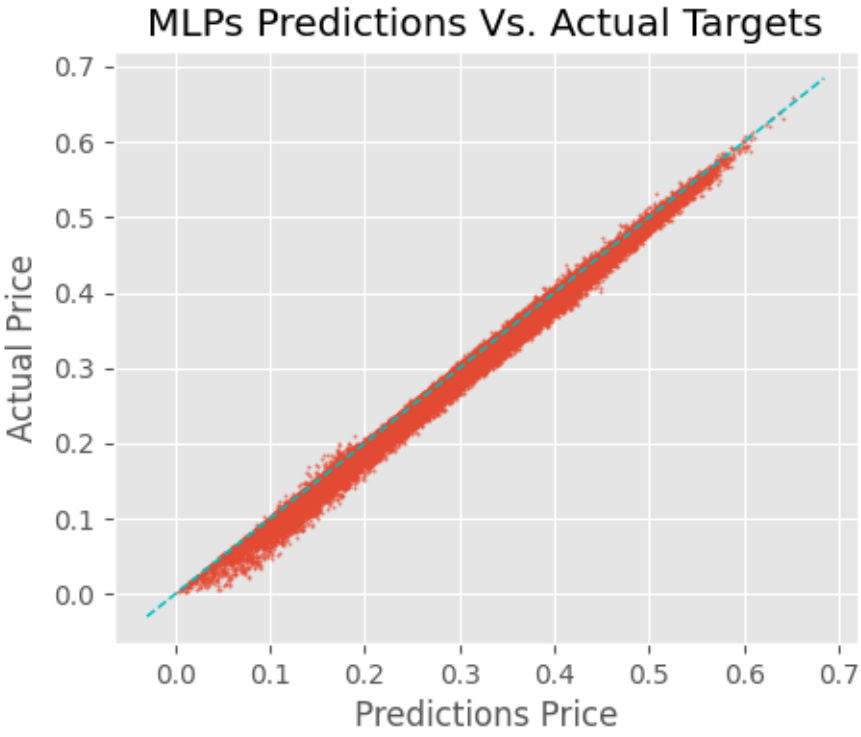


FIGURE D.6: MLP performance for long maturity data set on American put on minimum of two stocks

Bibliography

Andersen, Leif and Mark Broadie (2004). "Primal-Dual Simulation Algorithm for Pricing Multidimensional American Options". eng. In: *Management science*. Management Science 50.9, pp. 1222–1234. ISSN: 1526-5501.

Björk, Thomas (2009). *Arbitrage Theory in Continuous Time*. Third edition. Oxford.

Björk, Tomas (2019). *Arbitrage Theory in Continuous Time*. eng. Oxford Finance Series. Oxford: Oxford University Press. ISBN: 9780198851615.

Black, Fischer and Myron Scholes (1973). "The Pricing of Options and Corporate Liabilities". In: *The Journal of Political Economy* 81.3, pp. 637–654. URL: http://www.jstor.org/stable/1831029 (visited on 02/09/2020).

Boyle, Phelim P., Jeremy Evnine, and Stephen Gibbs (1989). "Numerical Evaluation of Multivariate Contingent Claims". eng. In: *The Review of financial studies* 2.2, pp. 241–250. ISSN: 0893-9454.

Buffett, Warren (2002). "Berkshire Hathaway's (BRK.B) annual letters to shareholders". In: URL: https://www.berkshirehathaway.com/letters/2002pdf.pdf (visited on 06/23/2020).

Clément, Emmanuelle, D. Lamberton, and Philip Protter (Sept. 2001). "An analysis of the Longstaff-Schwartz algorithm for American option pricing". In:

Cox, John and Mark Rubinstein Stephen Ross (1979). "Option princing: A simplified approach". In: *Journal of Financial Economics* 7, pp. 229–263.

Ekvall, Niklas (1996). "A lattice approach for pricing of multivariate contingent claims". In: *European Journal of Operational Research* 91.2, pp. 214–228. URL: https://EconPapers.repec.org/RePEc:eee:ejores:v:91:y:1996:i:2:p:214-228.

Elliott, Robert J. and P. Ekkehard Kopp (1999). *Mathematics of financial markets*. eng. Springer finance. Berlin: Springer. ISBN: 0387985530.

Ferguson, Ryan and Andrew Green (2018). "Deeply Learning Derivatives". eng. In:

Gaspar, Raquel M, Sara D Lopes, and Bernardo Sequeira (2020). "Neural Network Pricing of American Put Options". eng. In: *Risks (Basel)* 8.3, pp. 73–. ISSN: 2227-9091.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Hirsa, Ali, Tugce Karatas, and Amir Oskoui (2019). "Supervised Deep Neural Networks (DNNs) for Pricing/Calibration of Vanilla/Exotic Options Under Various Different Processes". eng. In:

Hull, John C. (2017). *Options, Futures, and Other Derivatives*. Vol. Ninth edition. Pearson Global Education.

Johnson, Herb (1987). "Options on the Maximum or the Minimum of Several Assets". In: *The Journal of Financial and Quantitative Analysis* 22.3, pp. 277–283. URL: https://www.jstor.org/stable/2330963?seq=1.

Kim, Lee, and An (Feb. 2020). "Real Option Valuation of the R&D Investment in Renewable Energy Considering the Effects of the Carbon Emission Trading Market: A Korean Case". In: *Energies* 13, p. 622. DOI: 10.3390/en13030622.

Kohler, Michael, Adam Krzyżak, and Nebojsa Todorovic (2010). "PRICING OF HIGH-DIMENSIONAL AMERICAN OPTIONS BY NEURAL NETWORKS". eng. In: *Mathematical finance* 20.3, pp. 383–410. ISSN: 0960-1627.

Lapeyre, Bernard and Jérôme Lelong (2019). "Neural network regression for Bermudan option pricing". eng. In:

Liaw, Richard et al. (2018). "Tune: A Research Platform for Distributed Model Selection and Training". In: *arXiv preprint arXiv:1807.05118*.

Longstaff, Francis A. and Eduardo S. Schwartz (2001). "Valuing American Options by Simulation: A Simple Least-Squares Approach". In: *The Review of Financial Studies*.

MacKay, David J. C. (2018). *Information theory, inference and learning algorithms*. eng. Repr. with corr. Cambridge University Press. ISBN: 0521642981.

McDonald, Robert and Anna Paulson (2015). "AIG in Hindsight". eng. In: *The Journal of economic perspectives* 29.2, pp. 81–106. ISSN: 0895-3309.

Merton, Robert C. (1973). "Theory of Rational Option Pricing". In: *The Bell Journal of Economics and Management Science* 4.1, pp. 141–183. URL: http://links.jstor.org/sici?sici=0005-8556%28197321%294%3A1%3C141%3ATOROP%3E2.0.CO%3B2-0 (visited on 05/14/2020).

Ouwehand, P. (2006). "PRICING RAINBOW OPTIONS". In:

OVERHAUS, MARCUS et al. (2007). *Equity Hybrid Derivatives*. eng. 1st ed. Wiley finance series. Hoboken, N.J: Wiley. ISBN: 0471770582.

Savine, Antoine (2019). *MODERN COMPUTATIONAL FINANCE AAD AND PARALLEL SIMULATIONS*. eng. Wiley. ISBN: 978-1-119-53945-2.

Shiryaev, Albert and Goran Peskir (2006). *Optimal Stopping and Free-Boundary Problems*. eng. Lectures in Mathematics. ETH Zurich. Basel: Springer Basel AG. ISBN: 3764324198.

Tsitsiklis, J.N and B van Roy (1999). "Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives". eng. In: *IEEE Transactions on Automatic Control* 44.10, pp. 1840–1851. ISSN: 0018-9286.